Realistic Virtual Cuts

Laursen, Lasse Farnung

Publication date:
2012

Document Version
Publisher's PDF, also known as Version of record

Realistic Virtual Cuts

Lasse Farnung Laursen

Kongens Lyngby 2012
IMM-PHD-2012-270

Technical University of Denmark
Informatics and Mathematical Modelling
Building 321, DK-2800 Kongens Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk

IMM-PHD: ISSN 0909-3192

Abstract

Pigs and pig meat are major sources of income for Denmark. As one of the country's primary exports, it is no wonder that Denmark strives to maintain its competitive edge in the meat market. As part of an on-going effort to lower costs and maintain high standards, X-ray computed tomography (CT), along with image analysis, is being deployed in Danish abattoirs. The data made available from scanning pig carcasses paves the way for new means to optimize the production process.

This thesis is concerned with the development of a communication tool intended to make use of the aforementioned technology in the product prototyping process. In broad terms, the focus can be divided into two areas of focus: visualization and interaction.

Visualizing volume data, obtained via CT-scanning, is a common area of research within other areas of research, e.g. for medical applications. The availability of graphics processing units, and the subsequent programmability of the unit, has allowed for computationally heavy visualization algorithms to execute in real-time. Despite the flexibility of modern GPUs, their architecture still poses problems that require further study. The thesis presents research within the area of texture synthesis and data interpolation in an effort to create even more realistic volume data visualization.

The potential advantages provided by volume data, is exponentially expanded when we are free to interact with it. The food industry sees a significant benefit in volume interaction when concerned with product development. Product earnings projection, product specifications, and interactive training are just a few of the applicable areas. In this thesis we present an interaction method intended

for the commercial development of meat product prototypes. The interaction method is evaluated in a thorough usability study with eight voulenteer participants from the target user group.

This thesis presents technology and research which, combined with the advent of using CT in the abattoir, paves the way for new possibilities and advantages when designing meat product prototypes. I have no doubt that this is just the tip of the iceberg in regards to modernizing and optimizing the way animal carcasses are processed and handled before becoming consumer goods.

Resumé

Svin og svinekød er én af Danmarks største indkomstkilder. Som et af landets primære eksportprodukter er det ikke overraskende, at Danmark stræber efter at være førende på et konkurrencebetonet internationalt marked. Som et led i den igangværende indsats for at mindske produktionsomkostninger og opretholde høje produktstandarder er danske slagterier begyndt at anvende røntgen computer tomografi (CT) og digital billedbehandling. Data der opsamles via disse metoder, baner vejen for nye måder til at optimere produktionen.

Denne afhandling omhandler udviklingen af et kommunikationsværktøj, som gør brug af ovemstående teknologi i processen for udvikling af kødproduktprotoyper. Afhandlingens fokusområder kan deles i to større kategorier: visualisering og interaktion.

At visualisere CT data er allerede et stort forskningsområde i andre felter, fx til medicinsk anvendelse. Den udbredte integration af grafikhardware og dertilhørende fleksibilitet af hardwaren, har tilladt at man nu er i stand til at køre yderst krævende grafiske algoritmer i realtid. På trods af fleksibiliteten, som moderne grafik hardware tilbyder, medfører hardwarens begrænsede arkitetktur stadig problemer, der burde undersøges. Denne afhandling præsenterer forskning indenfor tekstursyntese og datainterpolation i bestræbelsen på at opnå mere realistisk volume-visualisering.

De potentielle fordele, som CT-data kan tilføre,vokser eksponentielt, når brugeren får mulighed for at interagere med dem. Der er betydelige fordele forbundet med volumedata-interaktion i forbindelse med produktudvikling set fra fødevareindustriens perspektiv. Indtjeningsprojektion på produktet, produktspecifikationer og interaktiv træning er kun nogle af de mulige anvendelsesområderne. Denne afhandling præsenteres en interaktionsmodel for at udvikle kommercielle kødproduktprototyper. Interaktions modellen evalueres gennem en omfattende

'usability test', hvor otte brugere fra målgruppen deltog.

Afhandlingen præsenterer teknologi og forskning, som sammen med anvendelsen af CT i moderne slagterier baner vejen for nye muligheder og fordele inden for udviklingen af kødproduktprototyper.

Jeg er ikke i tvivl om, at dette kun er toppen af isbjerget med hensyn til at modernisere og optimere måden, hvorpå svinekød bliver bearbejdet og håndteret, inden det sælges til forbrugerne.

Preface

This thesis was prepared at the Image Analysis and Computer Graphics group at DTU Informatics and submitted to the Technical University of Denmark (DTU), in partial fulfillment of the requirements for the degree of Doctor of Philosophy, Ph.D., in Informatics and Mathematical Modeling. The project was funded by DTU, the Danish Meat Research Institute [28] and the ITMAN Graduate School programme.

The work herein represents selected parts of the research work carried out in the Ph.D. period. The thesis consists of three research papers, one technical report, and an introductory part containing background information and an overview of the contributions.

The work was carried out in collaboration with the Danish Meat Research Institute of the Danish Technological Institute (previously of the Danish Meat Association) in Roskilde, Denmark. Part of the research was conducted at the IGARASHI Laboratory of the University of Tokyo in Japan, under the supervision of Takeo Igarashi. The project was supervised by Professor Bjarne Kjær Ersbøll, from DTU Data Analysis, and Associate Professor Jakob Andreas Bærentzen from DTU Informatics, and by Senior Researcher, Ph.D. Lars Bager Christensen, Danish Meat Research Institute.

Kgs. Lyngby, April 2012
Lasse Farnung Laursen

List of Papers

Listed here are the scientific publications prepared during the course of the Ph.D. program. The publications included in part II of the thesis are referred to by their respective chapter number.

Conference papers:

Ch. 8 [**74**] L. F. Laursen, J. A. Bærentzen, and B. K. Ersbøll. Anisotropic 3D texture synthesis with application to volume rendering. *International Conference on Computer Graphics, Visualization and Computer Vision, WSCG ; 19 : Plzen, Czech Republic*. 2011.

Ch. 10 [**75**] L. F. Laursen, H. Ólafsdótir, J. A. Bærentzen, M. S. Hansen, and B. K. Ersbøll. Registration-based Interpolation Real-Time Volume visualization. *Proceedings of the 28th Spring Conference on Computer Graphics*. 2012.

Ch. 11 [**70**] L. F. Laursen, J. A. Bærentzen, T. Igarashi, M. K. Petersen, L. H. Clemmensen, and B. K. Ersbøll. Pig Product Prototyper: Cutting interface design. *NordiCHI 2012 Proceedings*. (Submitted, April 2012).

Technical report:

Ch. 9 [**71**] L. F. Laursen, L. H. Clemmensen, J. A. Bærentzen, T. Igarashi, B. K. Ersbøll. Automatic Quality Measurement and Parameter Selection for Example-based Texture Synthesis.

Publications not directly referenced or outside the theme of the thesis are listed below:

[24] L. K. H. Clemmensen, and L. F. Laursen. Improving texture optimization with application to visualizing meat products. *Scandinavian Workshop on Imaging Food Quality, SWIFQ : Ystad, Sweden, 2011.*

[73] L. F. Laursen, and B. K. Ersbøll. GazeTrain: A case study of an action oriented gaze-controlled game. *The 5th Conference on Communication by Gaze Interaction - COGAIN 2009.*

Co-authors of the publications in Part II and their affiliations are listed below in alphabetical order.

- JAKOB ANDREAS BÆRENTZEN
  DTU Informatics, Technical University of Denmark, Lyngby, Denmark.
- LINE KATRINE HARDER CLEMMENSEN
  DTU Informatics, Technical University of Denmark, Lyngby, Denmark.
- BJARNE KJÆR ERSBØLL
  DTU Informatics, Technical University of Denmark, Lyngby, Denmark.
- MICHAEL SASS HANSEN
  DTU Informatics, Technical University of Denmark, Lyngby, Denmark.
- TAKEO IGARASHI
  IGARASHI Laboratory, The University of Tokyo, Tokyo, Japan.
- HILDUR ÓLAFSDÓTTIR
  DTU Informatics, Technical University of Denmark, Lyngby, Denmark.
- MICHAEL KAI PETERSEN
  DTU Informatics, Technical University of Denmark, Lyngby, Denmark.

Acknowledgments

First and foremost I would like to the people who have, spiritually, contributed the most to this thesis: my current and former colleagues of the Image Analysis and Computer Graphics Group at DTU Informatics. It has been a pleasure fostering new friendships and talking with people in the same predicament as one self. A special thank you goes out specifically to my former colleagues of the 'glass prison' on the first floor of Building 321. I'm also grateful for the generous anonymous candy donation by an undisclosed group (yes, you read right), dispite the fact that I rarely consume candy.

I wish to thank both my supervisors at DTU Informatics, Bjarne Kjær Ersbøll and Jakob Andreas Bærentzen, for their continued guidance, optimism and supervision throughout the project.

I would also like to extend my gratitude towards the entire staff Danish Meat Research Institute in Roskilde, and especially Lars Bager Christensen who also supervised this project from start to finish. I am grateful for his interest in the project and positive perspective on almost any conceivable situation.

Takeo Igarashi also deserves my appreciation and gratitude for hosting me during my external stay in Japan, at the University of Tokyo and associated IGARASHI User Interface Research Group. I also extend my thanks to Jun Kato, Nobuyuki Umetani, and Kenshi Takayama, for helping me out upon arrival, making me feel welcome and generally making my stay more pleasant.

I would also like to thank Camilla Himmelstrup Trinderup, Jacob Lercke Skytte, for helping me proof read this thesis and providing me with valuable and

thoughtful feedback. Thomas Hammershaimb Mosbech and John Mox deserve extra special attention for their — special — contributions and attention to detail. Line Katrine Harder Clemmensen is also deserving of my thanks for her aid in statistical analysis.

Last but not least, I want to thank my parents Ivy and Johs, as well as my brother Daniel and my sister Tina, for supporting me all throughout my life. My thoughts also go out to my aunt Nane for whom I hope a speedy recovery.

Contents

Part I

Summation

Chapter 1

Introduction

In 2011, approximately 21 million pigs were slaughtered in Denmark [34]. About 90% of the resulting product was exported to more than 100 countries all over the world, corresponding to almost 1.9 million tonnes of pig meat exported in the year prior to 2011. This collective export of pig meat amounts to 28.1 billion DKK annually [35], which accounts for almost half the value of the danish agricultural export [33].

Denmark is among the leaders within the pig meat industry and is known for its high standards of quality. However, according to industry experts, Denmark is no longer years ahead of its competitors, as was the case about a quarter of a century ago. With its high dependency on export, the danish pig meat industry is under constant pressure to remain competitive against rivaling pig exporting countries, such as Germany and Spain. The most difficult challenge facing the Danish pig meat industry is how to maintain the current high standards of industry safety, product quality and price, despite the higher expense associated with Danish production costs.

A break down of the cost/profit relationship of Danish production of pigs is illustrated in Figure 1.1. It is clear that the biggest costs covered involve the raw materials, followed by the production wages. The most straight-forward profitable move is to reduce the costs of the raw materials involved. However, in the Danish pig meat industry, a majority of the suppliers are also the owners

[40]. Thus, a reduction in raw material costs would not result in a net gain for the owners, and is therefore not appealing.

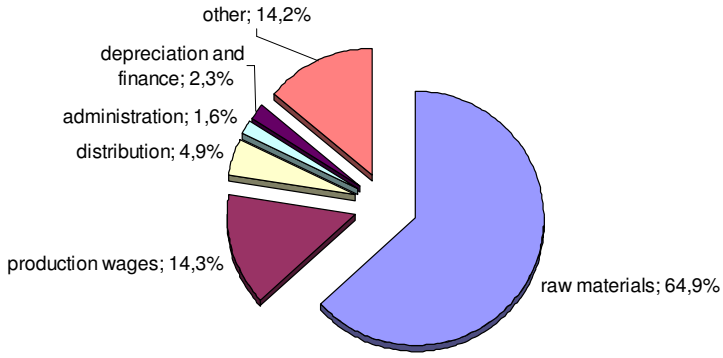

Figure 1.1: Cost/profit relation of the Danish production of pigs in 2009. The percentages reveal how large a portion of the profit from sold products was spent on the tasks required to produce them. For that particular year, the costs outweighed the profit and the chart sums to more than 100% as a result. Chart courtesy of Kjærsgaard [61].

The most appealing approach to minimize costs and increase profit, is to further optimize and innovate production, thereby remaining competitive without lowering wages or standards. The modernization of danish abattoirs using research and new technology [53] has formally been underway since the foundation of the Danish Meat Research Institute (DMRI) in 1954. Owned by the industry organization of Danish slaughterhouses, the institute has a close collaboration with the pig producers, focusing on developing methods and technologies for the safe and efficient production of meat products with high quality standards at competitive prices [28]. Merging with the Danish Technological Institute in 2009, the company's goals remain unchanged.

The recent introduction of X-ray computed tomography [58] in the danish abattoirs, for the purposes of more accurately estimating a pigs lean meat percentage, has opened up new possibilities for improvement in the pig product development cycle. In collaboration with DTU Informatics and related third parties, the Virtual Slaughterhouse project (VSH) was launched in 2006. The project demonstrated the practical application of X-ray CT along with image analysis in the context of an abattoir [32, 48, 116].

The availability of representative pig carcass volume data also paves the way to modernizing tasks surrounding the handling and production of pig meat

products. Specifically, the development, prototyping and selling of a new pig product to a prospective client. Although the Danish pig meat industry has played essential part in standardizing international practices with which a pig carcass is disassembled, there are still significant differences in the products being brought to the consumer market, in different parts of the world. Due to the natural biological variations among pigs as well as individual market demand, the number of viable consumer products is virtually endless. As a result, new products are continuously being developed to fit an ever changing market.

In broad terms, there are two kinds of potential product development scenarios:

- **New Product** — In this scenario, the prospective client usually contacts the pig meat producing company in order to buy a new type of product. The parties meet to discuss terms and resolve any potential ambiguities. If the terms are acceptable to both parties, a prototype batch of the discussed product is produced and an iterative development cycle is begun. At the end of the cycle, the final product is delivered in its entirety to the client. The development process from first contact until the final prototype has been created can take several weeks.

- **Repeated Product** — A pre-existing trade relationship is continued by either producing more of an already defined product, or iterating on the product with minor changes.

It is primarily the former of the two scenarios that is ideally suited to be improved upon, by leveraging the recently introduced CT-scanning technology. At the time of writing this thesis, the development of a new meat product for a prospective client is predominantly analogue and far from standardized. The communication is primarily verbal, with occasional use of reference images. Due to the natural biological variation in pigs and a lack of complete standardization in the industry, ambiguities occur. A single ambiguity not only means a waste of resources, both in terms of manpower and prototyped product, but also in terms of time. The development time from specifications to first prototype can take weeks, and portions of the process has to be repeated if problems occur.

This thesis advances the ideas and concepts from the VSH in a new direction. Previous work in the VSH has mainly centered its focus on the actual production line and associated tasks. This thesis is concerned with the iterative prototyping phase conducted prior to any actual production work. The thesis addresses existing communication issues through a developed communication tool and related technology.

1.1 Motivation and Objectives

The work presented in this thesis concerns research within the areas of visualization and human-computer interaction. The research is motivated by its applicability, in a commuication tool, intended to improve upon the development of new products in the meat industry. The design process behind this communication tool (application), is driven by a combination of industry knowledge, trial and error, and design heuristics. Significant design decisions are briefly noted in this section to motivate the topics of this thesis. The motivations behind the decisions, as well as the consequences are discussed in greater detail in the later chapters.

The recent introduction of CT-scanning technology in the abattoir has enabled the acquisition of volume data from pig carcasses. During product development, the anatomy of the pig serves as a makeshift product cutting blueprint. Rendering this volume data leads to a realistic depiction of the pigs anatomy, ideal for use during product development. To achieve real-time rendering performance, the tool leverages the computational power of modern graphics processing units (GPUs). Visualizing the virtual pig carcass is just the first step. The communication tool should provide functionality for interacting and cutting in the virtual pig carcass, to help define product specifications. The real world interaction between pig meat and knife was used as initial inspiration for interaction style provided by the tool. A trained butcher relies on the resistances exerted on the knife, as it makes its way through pig muscle, fat and bone. This tactile sensation can be simulated through a haptic feedback device. The Phantom Omni (©Copyright Sensable Technologies, Inc.) is an afforable haptic device, ideal as a primary candidate for haptic interaction. Interacting with volume data rendered via GPUs poses a number complex of challenges. Since most of the initial products developed from a pig carcass make use of planar cuts, which is also ideally suited for the hardware accelerated rendering, this cutting interaction was prioritized.

To sum up, the overall goal of this thesis is to research and develop a system for the purposes of visualizing and interacting with virtual pig meat. The thesis' main areas of focus are:

- **Visualization** — To research and apply methods for realistic rendering of volume data. Specifically, pig meat data.
- **Human-Computer Interaction** — To research and develop an interactive haptic-enabled cutting system for the purposes of simplifying and improving the existing communication process.

1.2 Thesis Outline

This thesis consists of two parts. The first part introduces and summarizes the work carried out during the Ph.D. program. The second part is comprised of the scientific publications prepared. Due to the self-contained nature of scientific papers, overlap occurs between the two parts.

Part I

Chapter 2 gives an introduction to the motivation and intentions of the work presented in the thesis, as well as drawing parallels to related projects.

Chapter 3 concerns itself with topics related to volume data in a primarily visualization context. The basics behind volume data representation, interpolation, rendering, interaction and visual enhancements are presented.

Chapter 4 gives an abstract introduction to design consideration and interfaces in general. Key principles behind formative usability testing are described, followed by the usability study conducted in association with the completed prototype interface. The study describes the entire process from initial planning and information gathering, to selected metrics, as well as the measured results.

Chapter 5 discusses future work of expanding the user interface, as described in chapter 4, as well as the necessity of integrating existing research regarding the parameterization of pig carcasses. The chapter concludes by briefly touching upon integration of the new technology into the current and future work flow in the meat industry.

Chapter 6 gives a summary of the results of part II and the main contributions of the thesis.

Chapter 7 contains a discussion of the contributions in relation to the objectives as well as future work. Finally, the chapter concludes the thesis with some closing remarks.

Part II

Chapter 8 presents a novel approach to improve volume rendering by using synthesized textures in combination with a custom transfer function. Existing knowledge is used to synthesize anisotropic solid textures to fit our volumetric data. As input to the synthesis method, we acquire high quality images using a 12.1 megapixel camera. The rendering pipeline is extended by creating a transfer function which yields not only color and opacity

from the input intensity, but also texture coordinates for our synthesized 3D texture. Thus, we add texture to the volume rendered images.

Chapter 9 describes automatic parameter optimization methods for example based texture synthesis. We cover research to directly estimate specific texture synthesis parameters, such as patch size and iteration convergence, based on input textures. We also examine various similarity measures and evaluate their effectiveness. The goal for each measure is to properly evaluate how well the resulting synthesis compares to the original input. A good similarity measure will enable the search for the optimal texture synthesis parameters by maximizing the quality of the synthesis as a function of parameters.

We apply the presented methods to a state of the art texture synthesis algorithm, namely the one proposed by Kopf et al [62]. It is easy to find a set of exemplars for which there is no single optimal set of settings. The results show a promising foundation for further research in establishing an automated optimal synthesis for a multitude of textures.

Chapter 10 addresses the issue of sparse anisotropic volume data. Interpolating anisotropic data using standard techniques results in suboptimal visual results. We present a novel approach for improving real-time volume data interpolation on a GPU. Our approach uses a pre-computed set of cross-slice correspondences to compensate for missing data.

We perform a qualitative analysis comparing sparse data sets derived from the ground truth. We investigate how the visual quality degrades as data becomes more sparse, testing the limits of the interpolation method. The method is ideal for reconstructing sparse data sets, as well as minimizing quality loss while scaling large amounts of data to fit on most mobile graphics cards.

Chapter 11 presents Pig Product Prototyper, an application intended to aid in the communication process between producer and retailer when developing new meat products for a constantly evolving market. The application interface allows the user to make planar cuts to a virtual pig formed from CT-scans of a real-world pig carcass. We perform a comparative study of two different controller interfaces for the application, one being a traditional mouse and keyboard input, and the other a six degrees of freedom haptic feedback device. The goal was to assess usability issues and overall usability for the target group concerning both interfaces.

The accurate depiction of pig anatomy can guide trained professionals to re-create standardized pig products. The results of the usability test with sales personnel do not lean significantly in favor of either interface, despite the participants expressing favor towards one interface. This stalemate carries a significance in regards to the more alien interface introduced to

the users. We describe the design process and observed user experience regarding the two interfaces.

1.3 Abbreviations

2D	Two Dimensional
3D	Three Dimensional
API	Application Programming Interface
CPU	Central Processing Unit
CSG	Constructive Solid Geometry
CT	X-ray Computed Tomography
DMRI	Danish Meat Research Institute
DOF	Degrees of Freedom
DTU	Technical University of Denmark
GLSL	OpenGL Shading Language
GPU	Graphics Processing Unit
GUI	Graphical User Interface
HCI	Human-Computer Interaction
HIP	Haptic Interaction Point
HU	Hounsfield Units
HUI	Haptic User Interface
LMP	Lean Meat Percentage
MRI	Magnetic Resonance Imaging
PCA	Principal Component Analysis
PET	Positron Emission Tomography
PPP	Pig Product Prototyper
RGB	Red, Green, Blue
VHIP	Virtual Haptic Interaction Point
VSH	Virtual Slaughterhouse

Chapter 2

Background

This chapter elaborates on the motivations mentioned in Section 1.1, by presenting background information related to the topics of the thesis.

2.1 Evolution of the Modern Abattoir

As emphasized in the introduction in chapter 1, the danish pig meat industry is a principal cornerstone of Denmark's exports, and is under constant pressure to evolve in order to maintain its competitive edge in the global market. As with other big industries, the pig meat industry has followed common optimization practices such as centralization, automation, and digitization:

- **Centralization** — In the past, slaughterhouses were smaller and functioned more independently with related offices in the same building. As part of the on going optimization effort, these distributed slaughterhouses have been consolidated into bigger more modern abattoirs. Local offices are still present, but the centralization has led to a consolidation of most offices into a single main headquarters.
- **Automation** — Arguably inspired by the automobile industry [57], the assembly line process of slaughtering pigs has, and continues to, become

more automated to make the process more efficient. Danish labor is expensive, and is therefore constantly the subject of optimization. Previous projects in the VSH have directly contributed to the efforts aimed at automating the abattoir. Notable successes include automated carcass cutting, and deboning procedures [53, 57].

- **Digitization** — Digitization is a broad term, but in this context it refers to the application of technology as a means of creating a digital representation of a real world product, allowing for affordances not provided by the real world counterpart. In this case, the application of X-ray computed tomography to create a digital representation of a pig carcass allowing for non-destructive interaction. Previous projects have aimed at allowing a more realistic representation of the digital pig carcass [88], by modeling pig tissue deformation.

Digitization is a critical development which paves the way for the described advances in technology and interaction described in this thesis. In comparison to centralization and automation, the process of digitization has a less direct impact on the bottom line in a competitive industry, and instead serves as an — as of now — relatively unexplored venue for improvement. It is precisely this type of innovation that the Danish meat industry must spearhead, in order to remain competitive, without sacrificing neither quality, nor price.

2.2 Product Prototyping

Digitization is a natural next step in the evolution and standardization of the meat product development process. What makes the scenario presented in this thesis particularly unique is the fact that, at the time of writing, no formal structure exists for product prototyping. The product development process is dynamic and highly dependent on the relationship between the buyer and the seller. Consequently, the process is more suited to be supported by modern technology, rather than being replaced by it.

Prototyping is an inherently destructive process when concerning meat products. An entire pig carcass is potentially wasted to produce a single prototype, which may or may not suit the buyers needs in the end. Introducing more realistic product representation earlier in the development process, affirms the customers expectations with regards to the real product being produced. The chances for unexpected ambiguities are therefore automatically lowered.

It is important to note that the representation provided by any currently existing

computer system cannot hope to compare to the actual real-world product. However, compared to verbal communication or static images, a more dynamic interactive system will undoubtedly give a better impression of the product under development. Additionally, a virtual representation has a number of affordances and advantages that a real meat product cannot have. Key among these is the non-destructibility and easier handling.

As stated in Section 1.1, the main objectives of this thesis is to research both the visual and interactive aspects of the aforementioned digital product representation. The application for the development of digital products has been named the Pig Product Prototyper (PPP). It embodies the intentions of the Danish meat industry, as a tool used during the communication process of new meat product development. In general terms, it is a volumetric visualization tool which provides an interactive cutting interface.

The following two sections present perspectives on visualization and interaction related to the PPP, followed by a more abstract view of the potential advantages provided by digital meat products, loosely related to this thesis.

2.2.1 Visual Appearance

As stated in the previous section, realistic appearance is important in order to meet or alter the customers expectations. The more realistically the product is presented during the prototyping phase, the less chance there is of wasting valuable resources and time on developing a product which may not suit the customers needs in the end.

With the acquisition of their first CT scanner in 2004, DMRI [28] took the first big step towards digitizing meat products. The CT scanner was previously owned by a hospital, and while poor in resolution compared to todays modern CT scanners, it is a significant leap forward in technology in regards to applications in the meat industry. With this technology at hand, a more realistic virtual representation of a given pig product is within reach. The anatomy of a pig is proportionally sufficiently similar to that of a human, so it can easily be scanned, without any modifications, in the CT scanner.

Volume data obtained via a CT scanner yields density values, also known as Hounsfield units (HU), as the scanned pig carcass in Figure 2.1 shows. Density data lacks appearance information, which is why the pig carcass is without proper coloring. The data itself is usually anisotropic as a result of the process used to acquire it. Modern CT scanners can also yield isotropic data, but the process is more time consuming, and future integration of a CT-scanner in the

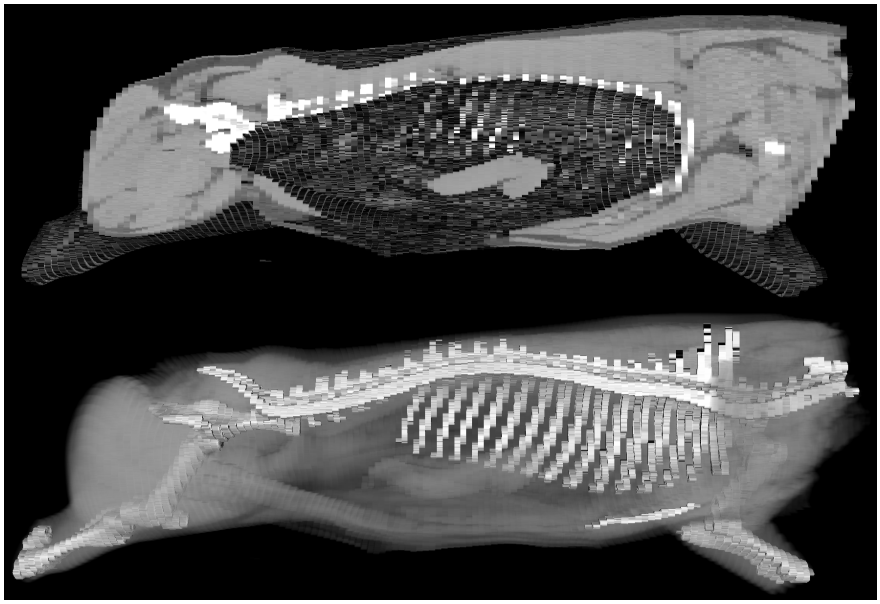

Figure 2.1: Visualized volume data representing half a pig carcass scanned by a CT scanner. Density values are scaled with the least to most dense being black and white, respectively. The top image shows an opaque rendering of the data with a cutting plane along the depth axis. The bottom image is a transparent rendering of the same data without any cutting plane. The distance in between measurements along the pig are spaced further apart than the other two axis, resulting in anisotropy.

slaughtering assembly line will require the scan to take no more time than any of the other machines, making isotropic scans less feasible.

While the visualized pig carcass is clearly recognizable to an expert, a lot is left to the imagination lacking both detail and color. Chapter 3 describes the properties of volume data in greater detail, and introduces the challenges in making this data both more realistic and visually appealing. The published contributions of this thesis concerning visual quality improvements of volume rendering is presented in chapters 8, 9, and 10.

2.2.2 Interaction

Digital objects are a strange phenomenon. They exist as concepts, metaphors, as well as counterparts of real world objects. The 'realness' of these virtual objects

continue to blur as multiple courts have, within the last decade, determined such objects to be legal property. In many aspects, this puts these virtual creations on equal footing with real world objects. However, it is beyond the scope of this thesis to debate how digital objects relate to real world laws from a legal perspective. The point is that whether or not something is real, depends on perception. Our perception plays a large role in how we interact with objects on a daily basis.

Traditionally visual and audial feedback have long been the primary means of computers to provide its users with feedback during interaction. But visual and audial stimuli affect just two of the five commonly recognized human senses. Interacting with any real world object usually requires touching it, and the utilization of this sense has seen a growing interest in research within the past few decades. One of the first researched applications stems from 1996 [59]. The earliest practical application haptic feedback dates even further back to early aviation history. Large aircrafts, not equipped with servo controls, would be subject to forces exerted on it by air pressure, when approaching a stalling position. The user (or pilot) would effectively be warned of the approaching stall by vibrations through the controls, caused by wind. Installing servomechanisms into these aircrafts provided benefits, but also removed the haptic side effect. As a result, the vibration of the simpler controls were simulated mechanically when the airplane's instruments measured an angle indicating an approaching stall.

Haptic interaction is also gaining ground within the medical industry. Although the most advanced medical interactive systems, such as the da Vinci or ZEUS [112] remote surgical machines, have yet to support haptic feedback, the potential for haptic feedback within the medical field is significant, and has been shown to positively effect the outcome of similar operations [14].

In this thesis, we explore a haptic enabled interface as a means of providing more realistic interaction during the production of pig prototypes. The introductory and technical aspects of both cutting and haptic rendering during interaction are presented in Sections 3.4 and 3.5 respectively. The overall design process of the interface implemented in the PPP as well as an introduction to Human-Computer Interaction form the content of Chapter 4. The published contribution concerning this topic is presented in Chapter 11.

Chapter 3

Volume Data

As the previous chapter explains, there is a significant motivation in the realistic depiction of pig carcasses in the PPP. Volumetric rendering is an ideal approach for achieving this realistic visual depiction. This chapter explains the fundamentals behind volume data and associated technical challenges of volumetric cutting and haptic rendering of volume data.

Real world objects are volumes, yet most graphics engines still rely on standardized triangle rendering algorithms. Triangles are the simplest geometric shapes with a surface and are computationally cheap to draw. Even more so, because all modern graphics hardware natively supports triangle rendering. But since triangles only represent surface area, their ability to realistically represent real world objects is limited. Volume data is a closer approximation to actual matter as it represents both what is on, as well as beneath the surface. The physical appearance of real world objects is highly dependent on more than just what's visible on the surface. An excellent example of this human skin. Slightly translucent, the human skin allows light to penetrate its surface and scatter beneath it, giving skin a very unique appearance. Accurate depiction of this appearance, is only possible by modeling what's beneath the actual surface of the skin. Thus, the need for volume data.

Volume data is defined as a three dimensional discretely sampled data set. Each point of data represents a sample at an infinitely small point:

$$f(\vec{x}) \in \mathbb{R} \; with \; \vec{x} \in \mathbb{R}^3.$$

Any position in our three dimensional space $\vec{x}$, yields a scalar value $f(\vec{x})$.

Figure 3.1 shows a simple conceptual illustration of a $3 \times 5 \times 3$ discreetly sampled data set.

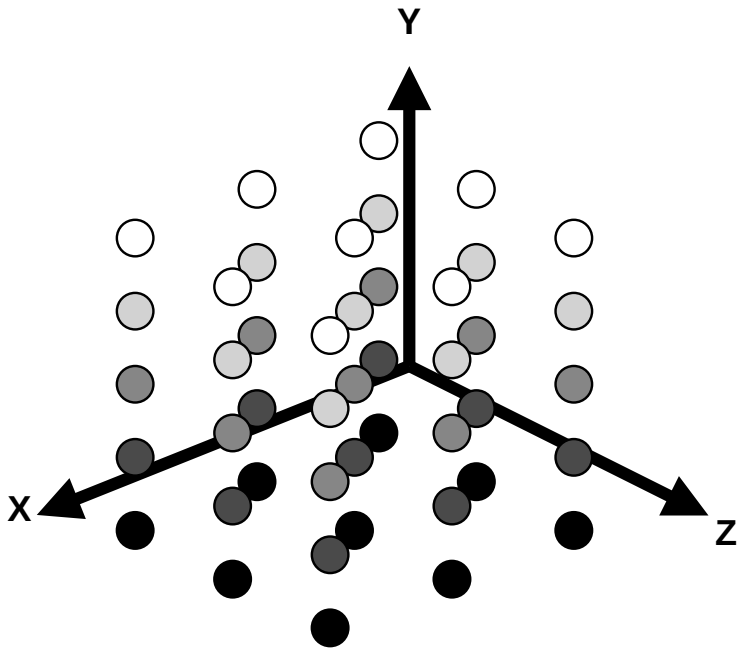

Figure 3.1: A simple $3 \times 5 \times 3$ representation of volume data. Each ball represents an infinitely small sampled point in space.

The individual points in volume data are often referred to as voxels and are usually visualized as simple geometric primitives, such as squares or spheres (as pictured in Fig. 3.1).

Since the first working prototype CT scanner developed in 1971 by Hounsfield et al. [12], there has been a constant drive to evolve and improve the techniques by which we obtain volume data. Today three dimensional data sets are obtainable through several means, e.g MRI, PET, or Ultrasound. CT scanning has naturally evolved increasing in effectiveness and resolution, as well as spawning a more focused and precise breed of scanners, referred to as Micro-CTs.

Hounsfield approached the problem of measuring a solid volume by simplifying the problem to measuring individual slices intersecting the volume. Modern CT scanners use more advanced software algorithms and process multiple slices as measurements are performed. These machines are referred to as helical or spiral CT machines, since they use a helical or spiral scanning pattern, respectively. The density in between measured slices depends on the time available and the required accuraccy. In most cases the measurement distance in between slices is higher, than the distance in between voxels in a single slice, in order to save time.

Single samples obtained via a CT scanner are measured in density, also known as Hounsfield Units [12]. A single sample represents the density measured on the Hounsfield scale. On this scale, air has a negative value of -1000, water density is measured as 0, and muscle measured at a value of 40. As an interesting side note, some of the first tissue to be measured by the prototype CT scanner was pig meat [12].

An overarching goal of this thesis is realistic rendering of volume data. Volume data is comprised of infinitely small points, measured from real world density. All the data between these infinitely small points is effectively missing. We need to apply methods with which to recreate this missing data, in order to render our volume data without visible sparsity. Techniques to achieve this are commonly known as interpolation.

3.1 Interpolation

Interpolation is the method with which we transform a discrete signal back into a continuous one. Given a set of samples, interpolation provides the means with which to recreate the data in between the samples. The optimal interpolation algorithm is highly dependent on the type of data that is to be interpolated. But in many cases, the type of data to be interpolated is not known in advance, so a general solution is often preferred.

Interpolation relies on samples to extrapolate the missing data. In broad terms, the two most significant elements of recreating missing data is how many of our samples to consider, and how to weigh each of their contributions. The process of recreating data via interpolation is referred to as reconstruction in signal-processing litterature, where filter kernels define how many samples to consider as well as their individual contribution. Figure 3.2 shows three common reconstruction filters: box, tent and sinc.

A continuous signal is reconstructed by convolving the discrete signal (i.e. the samples) with one of the reconstruction filters. The box filter corresponds to piecewise constant interpolation which results in a very sharp, but rather blocky reconstruction. The tent filter corresponds to linear interpolation and is commonly used as it provides an advantageous balance between computational complexity and acceptable reconstruction. The *sinc* reconstruction filter, illustrated in Fig. 3.2 (c), is the ideal reconstruction of the original signal.

However, there are two outstanding issues that invalidates the appeal of the *sinc* reconstruction filter. First, the *sinc* function is infinite and considers every sample to reconstruct the original signal. This is computationally infeasible and therefore not a viable solution. Second, according to sampling theory, if the continuous signal is band-limited with a cut-off frequency of v_s, it is possible to reconstruct the signal exactly, if the signal is sampled evenly at more than twice the cut-off frequency. But real-world data is generally not band-limited, so even given infinite resources, this approach would still not be able to reproduce the exact signal.

Instead, the tent filter is commonly used as it provides an advantageous balance between computational complexity and acceptable reconstruction.

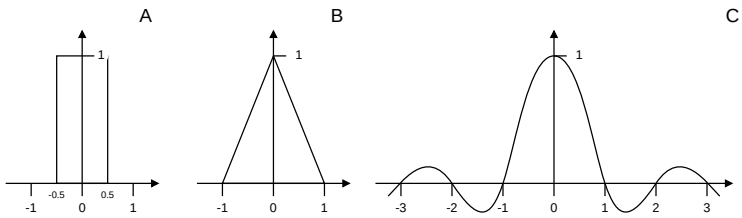

Figure 3.2: One dimensional reconstruction filters. A.) Box, B.) Tent, and C.) Sinc filter.

Interpolation using the tent filter is most easily described using a simple example. In a simple two dimensional coordinate space, we wish to determine the value of a point not located directly on one of our known sample points. In this case, it is point P, shown in Fig. 3.3 that we wish to reconstruct. Assuming we know the values of the surrounding four samples $(Q_{11}, Q_{12}, Q_{21}, Q_{22})$, determining the value of our unknown point P is quite easy.

We first linearly interpolate along the `X` axis yielding the values of R_1 and R_2 by calculating

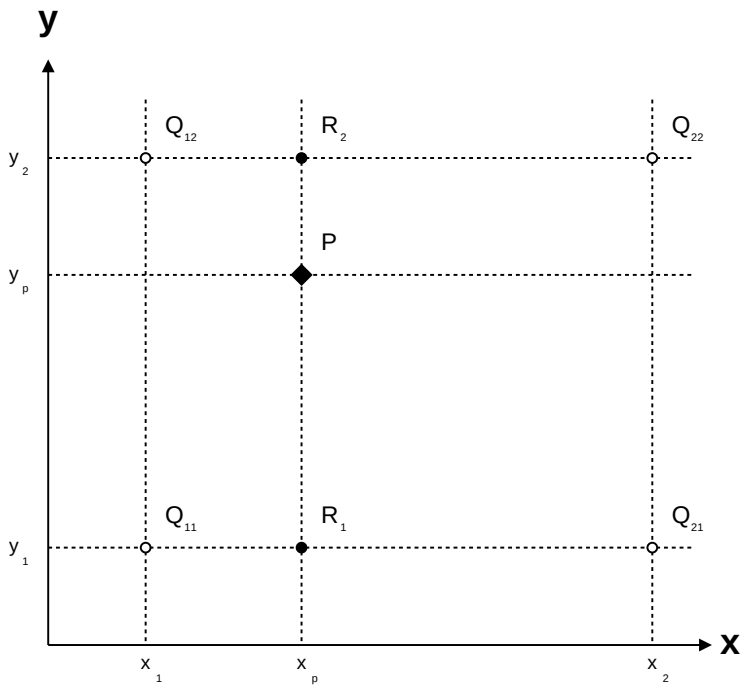

Figure 3.3: A typical interpolation scenario where we wish to estimate the value at point P, by interpolating the values of 4 close by known points $(Q_{11}, Q_{12}, Q_{21}, Q_{22})$.

$$f(R_1) \approx \frac{x_2 - x_p}{x_2 - x_1} f(Q_{11}) + \frac{x_p - x_1}{x_2 - x_1} f(Q_{21})$$

and

$$f(R_2) \approx \frac{x_2 - x_p}{x_2 - x_1} f(Q_{12}) + \frac{x_p - x_1}{x_2 - x_1} f(Q_{22})$$

respectively, where the function $f(\vec{v})$ yields the density value measured at point $\vec{v}$. Finally, the interpolated value at P is determined, by interpolating along the `Y` axis:

$$f(P) \approx \frac{y_2 - y_p}{y_2 - y_1} f(R_1) + \frac{y_p - y_1}{y_2 - y_1} f(R_2).$$

This approach, commonly known as bilinear interpolation, is easily extended into the third dimension, by performing the aforementioned operations on two axis aligned slices, and then linearly interpolating between the two resulting values. A simple example of bilinear interpolation applied in 2D space is shown in Fig. 3.4.

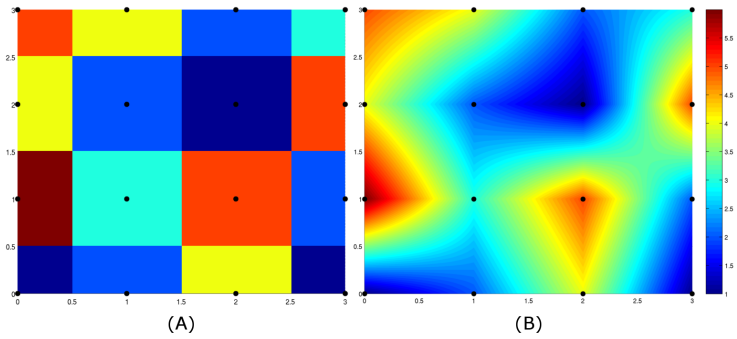

Figure 3.4: A simple 2D data set, with values ranging between 1 and 6 visualized using cool and hot colors respectively. On the left (A), the data is visualized using a box reconstruction filter. On the right (B), the same data is visualized using a tent reconstruction filter. Images adapted from the Wikipedia Commons Images by Berland [129].

Interpolating volumetric data is an essential tool for proper rendering. Even the densest set of volume data will appear sparse if viewed close enough. Linear interpolation is, in many cases, a more than adequate approach for reconstructing missing data. It is simple and also generally supported by hardware. However, the limitations of linear interpolation are especially noticeable when applying the method to anisotropic data. As already noted, volumetric data tends to be anisotropic in order to save time during scanning. A visual illustration of both the box and tent filter applied to volumetric data is shown in Fig. 3.5. Linear interpolation smooths the transition between data, but the result shows some considerable artifacts. The simple approach of using the tent filter results in improper weighted contributions by the surrounding voxels when estimating missing data. This problem is addressed in Chapter 10 where an advanced interpolation method is presented capable of real-time performance on modern graphics hardware.

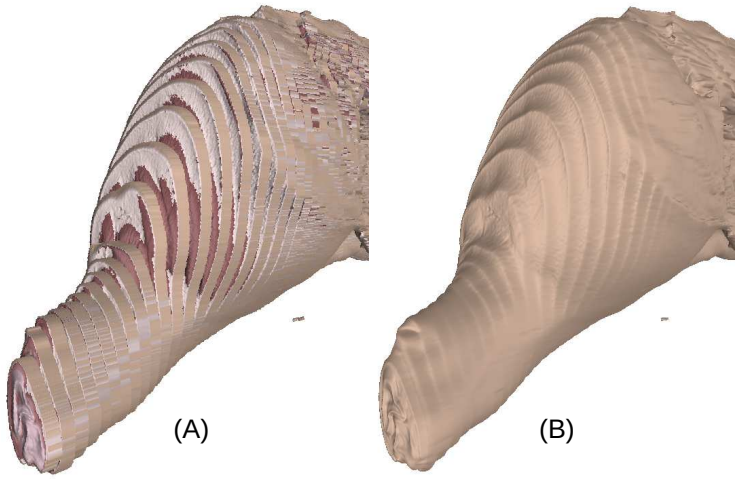

Figure 3.5: Anisotropic volume data, obtained by scanning a pig carcass. On the left, the volume is rendered using a nearest-value filter (box filter). On the right, the same volume is rendered using a tent filter.

3.2 Rendering

As established in the previous section, volume data consists of discretely measured samples in 3D. Interpolation allows us to reconstruct an approximation of the original continuous signal, critical for proper rendering of the volume itself.

Over the past two decades, GPUs have undergone significant changes in both design and operation, moving from a fixed non-programmable pipeline providing simple APIs, to completely programmable shaders providing advanced programming functionality and control structures [92]. Traditionally, 3D graphics rendering on a computer involves simple geometric shapes, such as triangles. Since a triangle is the simplest geometric shape that has a surface, the fixed graphics pipeline has evolved to primarily support this basic shape. Currently, no volumetric rendering primitives are supported on modern GPUs, so other means must be applied in order to render volumetric data.

Given that volumetric data sets are essentially individual slices through a dense volume, it seems only natural that early volume rendering techniques mimicked this, by rendering each slice on an individual transparent surface as a texture. The layered surface, when viewed from the proper angle, faithfully reproduces the image of the sampled volume.

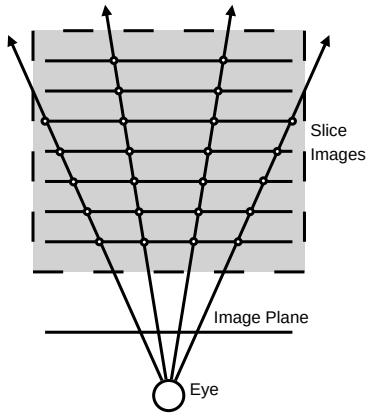

Figure 3.6: Illustration of an early volume rendering technique. Layered transparent slices with textures combine to form a visualization of the volumetric data. Note that the equal distance between image slices makes for an uneven sampling rate along the rays traced from the eye.

This early approach allowed for semi-hardware accelerated volume visualizations. However, the resulting images contain uneven sampling, due to each slice being uniformly spaced apart, as illustrated in Fig. 3.6. A more detrimental side effect of blending several transparent textured surfaces, is the fact that the most distant layers are inconsequential for the final result, leading to wasted computations and poor performance. Lacroute and Levoy mitigate a number of these issues in their notable shear-warp volume rendering algorithm [68]. However, the continued advancement of graphics hardware lead to other, more direct methods of real-time rendering, becoming feasible.

Ray tracing, originally pioneered for rendering purposes by Arthur Appel in 1968 [8], has proven itself to be ideal for executing on todays modern GPUs. The basic principles of raytracing are illustrated in Fig. 3.7, where a camera projects view rays through an image plane, on to a scene. Each ray corresponds to a single pixel on the image plane, and if the ray collides with an object from the scene, its color is calculated based on its position relative to existing light sources, in the scene.

With the advent of the modern graphics pipeline, illustrated in Fig. 3.8, both vertices and pixels can be assigned customized instructions (commonly referred to as shaders). Associating a single ray for every rendered pixel is an ideal fit for GPU accelerated volume rendering. Proxy geometry, such as a cube, is often used as a means of generating the required pixels for GPU raytracing. Krüger and Westermann [65] detail the GPU accelerated approach in a number of steps.

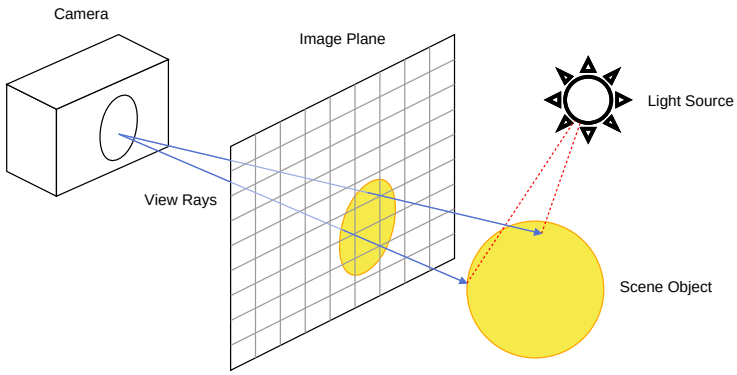

Figure 3.7: The essential elements of raytracing. A camera, from which view rays are projected through an image plane. Upon collision with a scene object, the angle to existing light sources is calculated and the final color value is calculated for that specific pixel on the image plane.

A simplified version more suited for the latest modern graphics hardware is described below, along with an illustration in Fig. 3.9:

- **Step 1 — Ray entry point determination** — The front faces of a volume bounding box are rendered into a 2D RGB texture, with the same size as the current viewport. Each color value is set to contain the 3D texture coordinates of the surface of the box. The result is a texture where each color value represents the first intersection point between the rays being cast, and the volume data. The values themselves are represented in coordinate space. The collision points for each ray with the proxy geometry is visualized in Fig. 3.9 (B).
- **Step 2 — Ray direction determination** — Similar to the first step, the backfaces are now rendered into a 2D RGB texture, with the same size as the current viewpoint. The collision points for each ray with the backfaces of the proxy geometry is visualized in Fig. 3.9 (C). Given the start and end point of each ray, calculating the direction of the ray is a simple matter of mathematical subtraction.
- **Step 3 — Ray traversal** — Having both the ray starting point and direction available, each ray is traced either until the end point of the ray is reached, or until the complete color of the related pixel is determined.

Using modern graphics hardware, supporting basic control flow, this entire process can be completed in two isolated rendering passes. One rendering the front

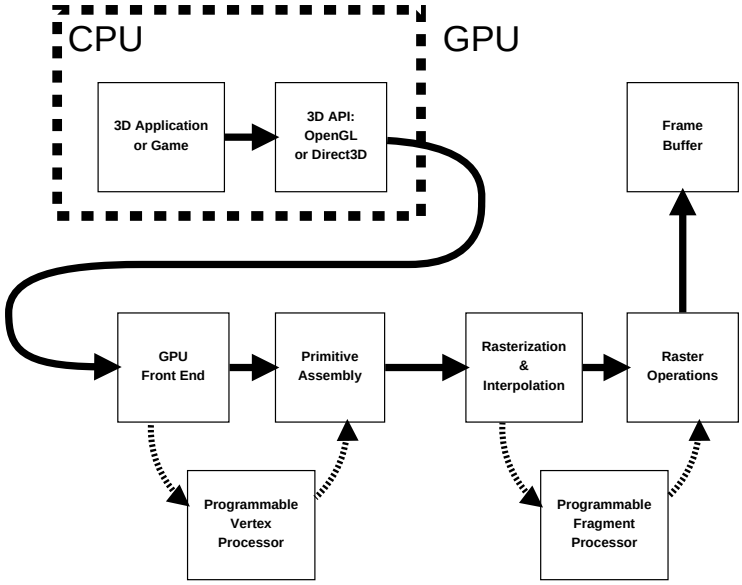

Figure 3.8: A simplified flowchart of the modern day graphics rendering pipeline.

of the volume bounding box, and another rendering the backside and iterating each ray from start to finish.

This style of direct volume rendering using the GPU is similar to general purpose programming, since the rendering pipeline is only used as a means to program the on-board fragment shaders. This somewhat unorthodox use of the graphics hardware leads to a few visual artifacts and caveats. Most of these are insignificant, except for proxy geometry clipping.

Prior to rasterizing the proxy geometry used in direct volume rendering in the graphics pipeline, visualized in Fig. 3.8, the geometry is clipped against two planes to prevent drawing unnecessary geometry. The front and back clipping plane, specifically. The only real concern for clipping the proxy geometry is the front plane, since it is conceivable that we would like to view the volume up close. If the proxy geometry is clipped, the shader will not be activated for any pixels and the volume data will not be rendered.

Fortunately, a simple yet effective change to the existing algorithm fixes this particular issue. Simply by filling the ray entry point texture with the eye position, before rendering the front faces into it. This will allow the near clipping plane to cut through the geometry and still visualize the volume. If the back

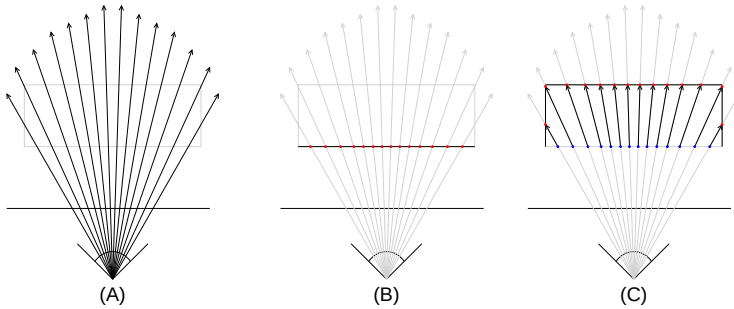

Figure 3.9: An illustration of how the approach described by Krüger and Westermann [65] uses a volume bounding box to perform ray casting on the programmable fragment processor. (A) A scene containing an image plane, along with a imaginary view rays and a volume bounding box. (B) The front side of the volume bounding box is highlighted to note how each of the imaginative rays cross the front face before (C) crossing the backside of our volume bounding box.

faces of the volume bounding box are clipped by the viewing plane, we would be looking past the volume and should not render anything regardless.

3.2.1 Transfer Functions

A question which has been left unaddressed so far, is how to determine the actual color of the volume data to render. Volume data does not contain any appearance modelling information, so the density value on its own is for all intents and purposes useless in deciphering the original color value. In most cases, the color is derived from pre-existing knowledge regarding the volume itself. Knowing that the volume data represents a pig carcass, it is possible to make accurate assumptions about which color should be associated with which density value (i.e. tissue type). Mapping voxel intensities from one domain ($\mathbb{R}$) to values used for visualization in another domain ($\mathbb{R}^4$) is typically called a transfer function [44].

Classifying types of transfer functions is not straight forward, as the approaches are almost as varied as the data they seek to enhance. The itemized list below categorize the more notable approaches by the most defining characteristic of each of the methods:

- **Linear transfer function** — The simplest transfer function simply maps a single density value ($\mathbb{R}$) to color values consisting of red, green, blue and

occasionally an additional value for transparency.

- **User input based transfer function** — The user based transfer function is similar to the linear transfer function, except that it allows a user to interactively modify the color values associated with various density values. This is useful for a number of volume visualizations where the densities of interest are not known in advance. Chourasia and Schulze [23] present an interface using opacity weighted histograms allowing the user to easily define transfer functions for high dynamic range data.

  It is important to emphasize that the usefulness of a user based transfer function is highly dependent upon the volume data being visualized and the methods used for proper segmentation. If the volume data is easily segmented using density thresholds, then the user can easily select areas of interest. Otherwise, more advanced techniques are required to separate areas of interest, such as interactive volume annotation.

  An example of semi-user input based transfer function is the visible human project [7], where a human carcass was cut into slices and photographed providing a 1-to-1 mapping of volume location and appearance.

- **Texture enhanced** — By using a piecewise constant function, it is possible to map a density value and voxel location directly to a texture best representing that density value. In order to provide the best visual appearance, the texture must be repeatable and conform to the dimensions of the volume data.

- **Texture based** — Volume data extracted from real world objects will often exhibit detectable patterns. Caban et al. [19] present a method for generating appearance information based on the texture surrounding a given voxel.

- **Wang Cubes** — Wang cubes [79] are an extension on an existing texture generation method called wang tiles [119, 120]. Ebert et al. [79] provide a framework for creating a large variety of non periodic illustrative 3D patterns and texture for use during volume visualization. The approach uses $8 \times 8 \times 8$ sized cubes stitched together according to a set of spatially oriented rules along with user input.

For the purposes of this thesis, a direct mapping between volume location and appearance information, similar to the visible human project [7], would be ideal. Especially given the availability of pig carcasses and the high interest in a representative volume. However, this type of mapping is very time consuming and beyond the scope of this thesis. At the time of writing, the only 1-to-1 mapping in existence for animals is for dogs [18].

Another appealing alternative would be the wang cubes, however given the small dimensions of the cubes involved in reconstructing the texture, it is questionable how well the results would recreate actual pig tissue.

In this work a texture enhanced approach was chosen to yield appearance information to the volume data representing pig carcasses. Specifically, a piece-wise linear function mapping density values to volumetric tissue textures.

It is worth noting that a persisting problem within volume rendering, is the surface density gradient. Regardless of the actual surface density of the object measured, interpolation will cause gradient based artifacts, e.g. muscle tissue appearing as skin. An example of this shown in Fig. 3.10.

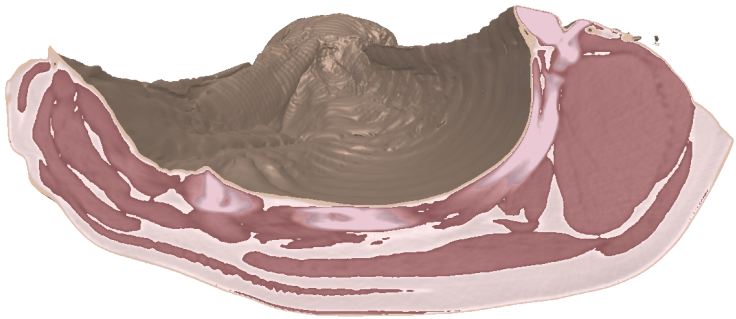

Figure 3.10: A screenshot of directly rendered volume data, using linear interpolation. This produces visual artifacts on the surface, giving the appearance of skin on the topside surface of the pig, when it actually consists of muscle and bone.

Tiede et al. [114] presents a more complex volume rendering approach, which calculates volume intersection more accurately and eliminates this visual artifact. Although the proper coloring of the pig carcass surface is relevant for the initial visual impression, it is of minor significance in the scope of this thesis for two reasons. First, realistic depiction of the pig carcasses internal anatomy is of prime importance. Second, the shape of the pig surface offers a significant amount of landmarks with which to properly recognize muscle and bone. Therefore, the visualized pig carcass in this work retains the appeareance of skin on the surface.

The developed transfer function is applied to muscle, fat and bone tissue. As previously mentioned, these textures need to conform to the same dimensionality as the volume data, i.e. three dimensions. Due to the lack of pre-existing volumetric textures of this kind, the problem is solved via synthesis. The following section presents a general introduction to notable texture synthesis approaches

and forms the basis for the contribution of this thesis to the field, presented in Chapter 8 and 9.

3.3 Texture Synthesis

Texture synthesis is the process of algorithmically creating a larger digital image based on a small user defined input sample. The areas of application are wide within computer graphics, and it therefore comes as no surprise that there is a significant amount of pre-existing research in this area.

Regardless of the approach or method applied the overall goal is the same. Given a small texture, we wish to generate a larger version. This task is made considerably more difficult by the fact that, not only does the larger version have to resemble the small sample, but it also has to avoid containing unsightly seams and reduce noticeable periodicity.

Textures are often divided into two categories: deterministic, and stochastic. Deterministic textures consist of a set of primitives along with a fixed spatial distribution (e.g., a brick wall). Stochastic textures are sporadic in nature and do not consist of easily identifiable primitives (e.g., bark, marble, sand). Real-world textures often consist of a combination of the two (e.g., woven fabric, wood boards, cobble stone).

A number of texture synthesis methods, categorized by the overall applied approach, is listed below:

- **Parametric** — Parametric approaches assume a certain distribution, often Gaussian, and generate textures based on this prior knowledge. Heeger and Bergen [52] present a method targeted at synthesizing stochastic textures.
- **Non-Parametric** — Non-parametric approaches do not assume any pre-existing distribution, but instead analyze the existing structure of the provided user input to determine and re-create a similar structure in the synthesized result. De Bonet [27] presents a method with which the original texture is analyzed, and based on the detected structure a larger version synthesized.
- **Pixel Based** — Pixel based methods are similar to non-parametric approaches in the sense that they also analyze structure, but in a more indirect fashion. The synthesized result is created by successively adding

new pixels based on surrounding neighbors. Wei and Levoy [127] apply a Markov Random Field as a structural element in their algorithm, while Harrison [51] uses an alternate approach, more focused on reproducing local spatial structure.

- **Patch Based** — Patch based methods apply the general principle behind pixel based methods on a grander scale. Instead of simply having one pixel's value determined by its neighbors, patch based approaches insert bigger sections of the original sample in an effort to successfully reproduce all the features found. Kwatra et al. [67] present an approach which determines the optimal portion of the original sample image to insert given a location in the image being synthesized.

- **Optimization Based** — Kwatra et al. [66] present an optimization based approach, that iteratively alters the composition of the synthesized texture. Starting from a random selection of pixels from the users input texture, the synthesized texture is divided up into several axis-aligned neighborhoods. Each neighborhood in the synthesized texture finds its best corresponding match in the original input texture, and the texture is iteratively changed to look more like these best-fit neighborhoods.

As stated in Section 3.2.1, the textures of the various types of pig tissue required for the creation of a customized transfer function must correspond to the dimensionality of the volumetric data. Although some of the methods presented above lend themselves to solid texture synthesis [52], the above cited research primarily attempts at recreating two dimensional textures. Within the past two decades, there has been a push toward 3D texture synthesis, based on many of the aforementioned approaches:

- **Parametric** — Ghazanfarpour and Dischler [39] present an approach which utilizes spectral analysis on a 2D user input sample, and extracts parameters for the purposes of generating solid textures.

- **Non-Parametric** — Wei [123] generalizes the approach based partially on Markov Random Fields to generate 3D textures.

- **Stereological** — Jagnow et al. [56] apply traditional stereological methods, concerned with extracting quantative information about 3D material from 2D samples, to generate solid textures.

- **Optimization Based** — Kopf et al. [63] expand upon the work pioneered by Kwatra et al. [66] by generalizing it to three dimensions. By comparing texture patches on axis aligned slices and accelerating the process, the method synthesizes plausible solid textures based on 2D input.

Out of aforementioned approaches, the texture optimization proved the most promising, which we have adapted for generating solid textures based on pig tissue.

A thorough presentation of this method as well as its specific application to pig carcasses is presented in Chapter 8.

3.4 GPU Accelerated Volume Cutting

During the development process of the PPP prototype interface, various cutting approaches were considered for viability in regards to direct volume rendering on the GPU. As previously explained in Section 3.2, direct volume rendering relies on proxy geometry to make heavy use of GPU shaders in order to achieve impressive visuals at real-time speed. Therefore, imposing cuts on the proxy geometry, would effectively impose the same cuts to the volumetric data.

At this point it is prudent to differentiate between incision-style cuts and complete cuts. Bruyns et al. [17] present a large body of work concerned with making custom incisions, that could be expanded to complete cuts, in polygon meshes. However, the polygon mesh (i.e. proxy geometry) used to render the volume data exists only as technical necessity to make use of a GPUs shader cores. Therefore, incisions imposed upon it would not work as intended. The incisions would effectively be applied to the surrounding proxy geometry and not the pig carcass volume data.

An appealing alternative is constructive solid geometry (CSG). It offers the flexibility of performing logical operations, reminiscent of set theory, to geometric shapes. It allows the subtraction of geometric shapes from the proxy geometry surrounding the volume data, and would result in correctly rendered visuals. Bernstein and Fussel [13] present a novel approach that overcomes the issues of calculation complexity and robustness that have normally plagued CSG algorithms. However, the high flexibility is not essential to the type of cuts prioritized for the PPP interfaces. Fully customized cutting can be achieved quite efficiently on volumetric data without the need for complex mesh partitioning algorithms, as detailed at the end of this section.

A viable compromise is the CSG algorithm presented by Stewart [110], which performs the intersecting calculations directly on the graphics hardware in image space. This means that the mesh is never actually partitioned or altered in any way, instead, each pixel has its depth set to the proper depth by the intersection geometric shapes. This effectively sidesteps the issue of robustness normally

plaguing CSG algorithms.

Figure 3.11 shows the algorithm intersecting the proxy geometry (in red), with two blue squares.

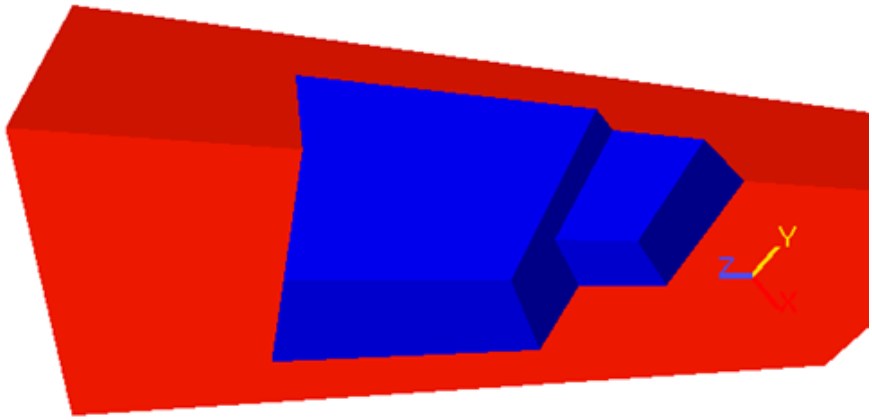

Figure 3.11: The frontfaces of the proxy geometry surrounding the pig carcass visualized in bright red, intersected by two smaller blue cubes.

Unfortunately, the algorithm (aptly named sequenced convex subtraction) was never intended to properly intersect concave objects, such as the backface of the proxy geometry, which results in artifacts as visualized in Fig. 3.12.

Modifications to the algorithm allowed it to produce proper results under certain conditions, but not all of the possible intersections using concave objects could be solved using the flexibility provided by the GPU to intersect objects in image space.

By simplifying the intersecting geometry to planes, the intersection algorithm could be implemented directly on the GPU rendering the pig carcass volume. The ability to represent individual planes using only four numbers: a three dimensional normal and one distance scalar, allows for the geometry to be easily handled on the GPU. Fig. 3.13 illustrates this representation of two different planes.

By calculating intersections of each of the rays required to visualize the volume,

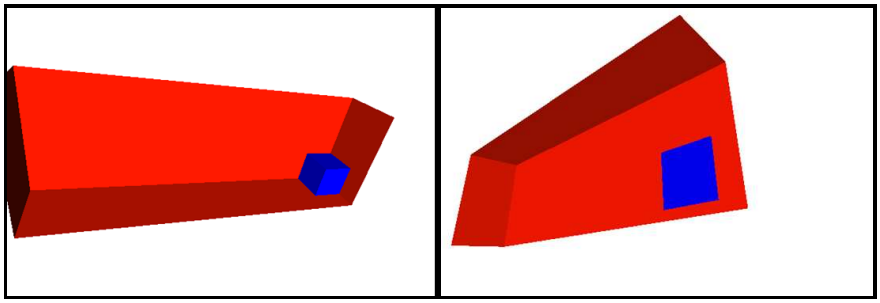

Figure 3.12: The backfaces of the proxy geometry surrounding the pig carcass visualized in bright red, intersected by a single small blue cube. On the left, a modified sequenced convex subtraction algorithm correctly produces the intersection between the two objects. On the right, a failure case missing partial geometry.

described in Section 3.2, with the introduced cut planes, it is relatively straight forward to only render the remaining volume, while optimizing the rendering process.

For optimal performance, it is necessary to recompile the shader programs executing in the GPU in real-time, for every cutting plane introduced. A shader optimized version of a standard line-plane intersection algorithm [130] is used to determine the start and ending point of each of the rays traversing the volume.

In addition to the planar cuts, a binary mask is applied to the volume which allows the user to cut individual voxels. This functionality allows for completely customizable cuts, but was not prioritized for the primary interactive functionality provided by the PPP. It is also important to note that providing custom cutting control to the user is, at best, a questionable design choice. This topic is further explored in Chapter 5.

3.5 Haptic Rendering

Haptic refers to the sensation of touching, and is arguably the third most used of the classical human senses, in computer applications. With its origins in aviation history, with a vibrating flight-stick alerting the pilot of an oncoming stall, it has found stable ground in the video game industry [20]. It was widely popularized by Sony's Playstation Dualshock ® Controller, as well as several driving and flying simulation input devices. Although it has yet to be adopted as a standard in the medical industry, recent research [14] indicates that operational procedures stand to benefit from its application.

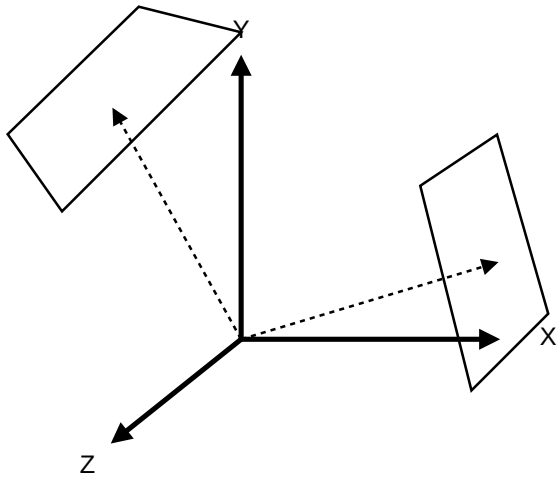

Figure 3.13: A visualization of two planes represented via a scalar applied to the planes normal, yielding its final location and orientation in world space.

Given the already significant load of information levied on the users two primary senses, sight and hearing, it is not surprising that there is a high interest in leveraging information on an alternate sense. This motivation has lead to the development of haptic devices such as the Phantom Omni [94], visualized in Fig. 3.14, the same controller utilized during the development of this thesis.

The common senses humans posses, are sensitive to different frequencies of input. For example, images changing at a frequency of 24 hz, is enough to convince the human eye that the visual input is animated, rather than static. Haptic feedback, on the other hand, requires a much higher frequency to maintain sensable stability for humans. The update frequency required for proper universal haptic feedback is often set to 1000 hz [76]. That is, the force feedback provided by the haptic device (such as the Phantom Omni) has to be calculated and applied 1000 times during a single second in order for users not to feel a significant discontinuity. Thus, any algorithm intended for haptic device rendering must often conform to this real-time 1000 hz standard.

As established earlier in this thesis, the data we are interested in interacting with, consists of volumetric data from a CT scan of a real pig carcass. A number of approaches exist for the purposes of haptically interacting with volume data. In broad terms, they can be categorized into the following groups:

- **Alternate Proxy Representation** — Although not a method of directly interacting with volume data, it is still important to present the

Figure 3.14: The Phatom Omni haptic feedback device © Copyright Sensable Technologies, Inc. Provides the user with six degrees of freedom, and is capable of giving haptic feedback on three of the six axes. The input device also features two buttons on the pen. Photo used with permission from Sensable Technologies, Inc.

possibility of converting the current representation of data into another viable alternative, such as a polygon mesh, using a mesh generation algorithm (e.g. marching cubes algorithm [78]). The advantage of a mesh is that the surface is well defined and the Phantom Omnis API [55] natively supports mesh based haptic interaction.

- **Direct** — A direct representation of the input device is used as a basis for interacting with volume data. Petersik et al. [98] partially base their haptic rendering algorithm on this approach, where multiple sample points on the surface of the virtual representation of the tool are used to calculate the forces acting on it. Figure 3.15 illustrates an example of this force calculation.

- **Indirect** — The forces imposed upon the haptic feedback device are generated indirectly by using a proxy object. The actual haptic feed back device cursor position, is referred to as HIP (Haptic Interaction Point), and travels freely within and out of the virtual objects. A proxy object, referred to as VHIP (Virtual Haptic Interaction Point), is subject to the constraints provided by the various interactable objects in the virtual scene. The difference in position between the VHIP and HIP form the ba-

sis for the force enacted upon the haptic feedback device. An illustration of this concept is provided in Fig. 3.15.

Palmerius [95] presents extensive work within the field of generating haptic feedback from volumetric data, and differentiates further between various types of haptic interaction algorithms.

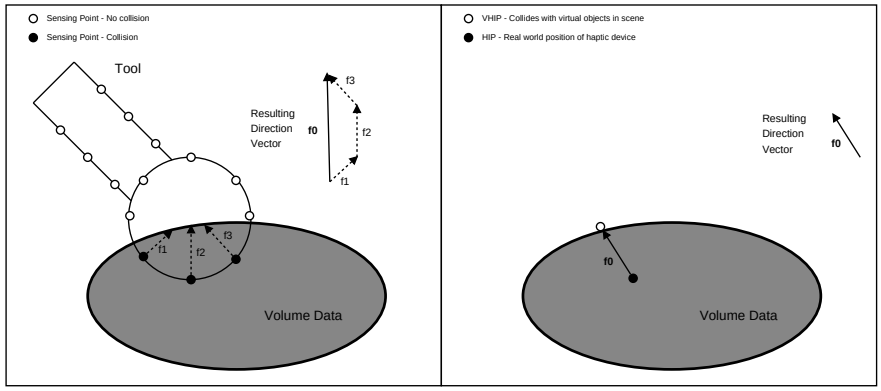

Figure 3.15: A visual illustration of two different approaches for calculating haptic feedback based on volume data. On the left, the direct method comprised of several sensory points which together yield a complete directional force vector. On the right the more indirect method of having a proxy object interact with the virtual volume and deriving a force vector between the difference in location of the "real" and the proxy object.

Since the cutting algorithm detailed in the previous section acts upon the volume data directly, it would be counter productive to implement a haptic rendering algorithm that requires a mesh representation.

The aforementioned direct approach could potentially yield the most realistic haptic feedback, assuming that multiple sense points on a tool representing the real-world pen on the Phantom Omni, would generate the resulting force direction. However, the Phantom Omni only provides haptic feedback on three of the six degrees of freedom offered by the Phantom Omni. Therefore, this direct style of haptic feedback using multiple sense points is unnecessarily complex for the type of force feedback the Phantom Omni is capable of providing.

Indirect rendering is well suited for the type of haptic feedback the Phantom Omni is capable of simulating. Notable works within indirect haptic rendering algorithms include, as previously noted, Palmerius [95] who provides a thorough in-depth analysis of haptic rendering using proxy objects. Menelas et al. [85] present a novel algorithm for rendering haptic feedback based on high frequency

volume data.

However, the complexity of both these algorithms are unnecessary given the relatively well defined low-frequency isosurface of the pig carcass. Chen et al. present a straight-forward algorithm based on intermediate isosurface representation [21]. A general introduction to the haptic rendering algorithm used for the PPP, based on the approach by Chen at al., is presented in the following subsection.

3.5.1 Isosurface Haptic Rendering

As briefly explained in the previous section, the direct haptic rendering method relies on a set of points, VHIP and HIP to describe the haptic feedback generated by the volumetric data. The location of the HIP always corresponds to the actual input position of the input device, which in the case of this thesis, is the input position of the Phantom Omni. The VHIP attempts to locate itself at the same location as the HIP, but is constrained by the volumetric data present in the scene.

By constraining the VHIP through collisions with objects rendered in the scene, it is possible to use a spring force, calculated in between the VHIP and HIP, as haptic feedback.

There are four different scenarios as far as the position of the VHIP and HIP are concerned, in relation to the volumetric data:

- **HIP outside of volume, VHIP outside of volume** — If the HIP is located outside of any volume data, determined by the isosurface threshold, then the VHIP should be set to the same location.
- **HIP outside of volume, VHIP inside of volume** — This is usually the scenario that occurs briefly after the HIP has any area occupied by the the volume data. Like the scenario above, the VHIP should be set to the same location as the HIP.
- **HIP inside of volume, VHIP outside of volume** — This scenario occurs when the user begins interacting with the volume. In this case the isosurface in between the VHIP and HIP is estimated, upon which the VHIP is then placed.
- **HIP inside of volume, VHIP inside of volume** — Given a volume with a well defined low frequency isosurface, this scenario rarely occurs.

If it does, it's possible to either seek the nearest surface for the VHIP to relocate to, or leave the behavior undefined.

The four scenarios described are implemented and illustrated via the psuedocode, shown in the listings box 3.1. As the code illustrates, the fourth scenario is left undefined as it practically never occurs given the pig carcass used in the PPP.

Listing 3.1: Surface Interaction Heuristic

```
if (VHIP.is_on_surface) {
    if (density_at_location(HIP) ≤ isolevel_density_threshold) {
        VHIP.is_on_surface = false;
        } else {
        VHIP.is_on_surface = true;
    }
} else {
    if (density_at_location(HIP) > isolevel_density_threshold &&
    density_at_location(VHIP) ≤ isolevel_density_threshold) {
        VHIP.position = estimate_isosurface_between(HIP, VHIP);
        VHIP.is_on_surface = true;
    }
}
```

If the VHIP is determined to be on the surface of the volume data (according to the third scenario), then it is necessary to perform some additional steps, in order to allow it to move along the isosurface. A simple technique employed by a number of haptic rendering algorithms is to use an intermediate representation, such as a plane, to move the VHIP along. The plane is defined using the current location of the VHIP, and the isosurface gradient of the volume data as its normal.

Figure 3.16 illustrates this approach. A virtual surface plane is estimated, upon which the vector from the VHIP to the HIP is projected. The VHIP is then moved a small increment along that vector. Moving the VHIP risks it leaving the isosurface, so we have to determine whether or not the VHIP has moved beyond the level of tolerance in regards to the isosurface. Since the isosurface of the pig carcass is relatively well defined, we can iteratively move the VHIP closer until it is on the isosurface. The pseudo code for this procedure is shown in listings 3.2.

Listing 3.2: Surface Movement Heuristic

```
if (VHIP.is_on_surface) {
    vec3 VHIP_to_HIP_surfaceVector = estimate_isosurface_vector(↩
        ↪ HIP, VHIP);
```

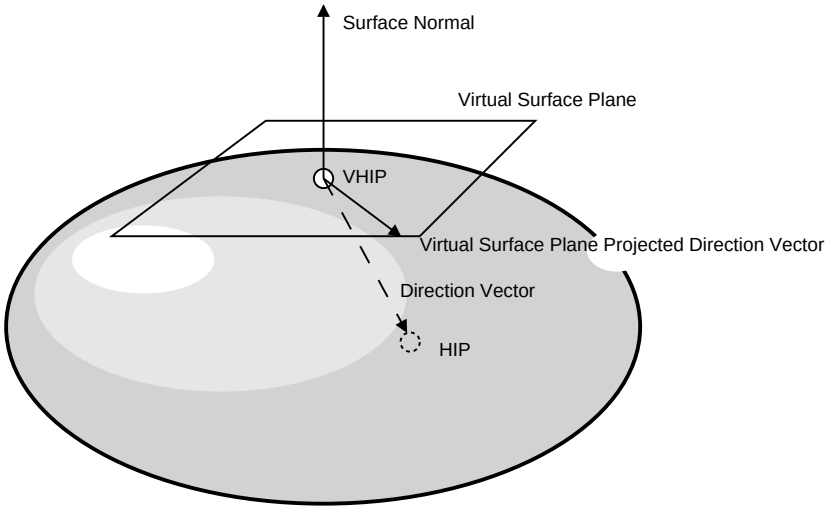

Figure 3.16: An illustration of the estimation of a virtual plane along the isosurface of spherical shaped volume data. The VHIP is positioned on the surface of the volume, while the HIP is inside the volume itself. The normal of the volume data at the location of the VHIP is estimated, yielding the gradient of the plane. The vector in between the VHIP and HIP is projected onto the virtual plane and used for a small incremental movement of the VHIP.

```
VHIP.position = VHIP.position + VHIP_to_HIP_surfaceVector;

// If VHIP has moved too far from the isosurface
if (density_at_location(VHIP) ≥ ↩
    ↪ isolevel_density_distance_tolerance) {

    vec3 surfaceGradient = estimate_surface_gradient(VHIP);

    // Is VHIP too deep or too shallow?
    if (density_at_location(VHIP) < isolevel_density_threshold↩
        ↪ ) {
        // Too shallow

        for (int i = 0; i < 100; ++i) {
            VHIP −= 2.0 * isolevel_density_distance_tolerance ↩
                ↪ * surfaceGradient;

            if (density_at_location(VHIP) > ↩
                ↪ isolevel_density_threshold)
                break;
        }

    } else {
        // Too deep
```

```
                for (int i = 0; i < 100; ++i) {
                    VHIP += 2.0 * isolevel_density_distance_tolerance ↩
                        ↪ * surfaceGradient;

                    if (density_at_location(VHIP) < ↩
                        ↪ isolevel_density_threshold)
                        break;
                }

            }
        }
    } else {
        VHIP.position = HIP.position;
    }
```

Expanding the algorithm to also perform properly when creating planar cuts through the volume data, is relatively straight forward. The `density_at_location` function referenced in the pseudo code, is modified to return a density corresponding to air, if the HIP is located on the "cut" side of any planar cuts. The function returns a steady gradient aligned with the plane, so the isosurface is established to be exactly on the visual plane, allowing the user to feel the planar surface of the cut volume.

Having established the proper position of the VHIP and HIP for haptic rendering, it is straightforward to calculate the spring force to apply:

$$springForce = gain * (VHIP - HIP) \tag{3.1}$$

The equation yields a force vector directly applicable to the haptic feedback device, allowing for quick, simple, and direct haptic feedback.

Chapter 4

Human Computer Interaction

As the term Human-computer interaction (HCI) indicates, it is all about the interaction between people and computers. HCI is an intersection of several different fields of study, for the purposes of establishing best practices and standards in shaping and designing the interaction between people and computers. The term also embodies the evaluation of the resulting interfaces and interaction patterns established by these practices. In relation to this thesis, HCI obviously plays a significant role given its focus on interaction. Both the design and evaluation of the PPP interface is firmly within the realm of HCI.

Section 4.1 gives an abstract introduction to design considerations and interfaces in general. The remaining sections describe key principles behind formative usability testing, as well as the usability study conducted in association with the completed prototype interface. The study describes the entire process from initial planning and information gathering, to selected metrics, as well as the measured results.

4.1 Interface Design

Interface design is communication design. Its existence is not restricted to reside between man and machine, but also in between people. The existing inter-human interface of visual and audial cues has evolved over thousands of years and would arguably be impossible to replace. Interfaces in between computers are often invisible as communication occurs with little or no visual or audial cues whatsoever. But despite the lack of these tells, communication exists regardless. The point is, communication is everywhere, and by a broad definition, so are interfaces. The PPP interface crosses the boundary between humans and computers, which is therefore the focus of interface design in this thesis.

An often reoccurring term in relation to interfaces is 'transparency'. The term often relates to the ease of communication in-between the two parties facilitating the communication. In a sense, it conveys the notion that the interface is actually a barrier and the less we notice it when communicating with the other party, the better. While this might be true in theory, it would require that both parties to be able to interpret each others responses and intentions perfectly. In practice between man and machine, it would require the machine to read and interpret man's intention without exchanging a word. Effectively requiring the machine to read minds. While fascinating in principle, this concept is currently beyond the grasp of human ingenuity and we are forced to examine the situation where an interface is more opaque than transparent.

Having established that a discernible interface is unavoidable, we must strive to implement the best interface possible. But what makes a good interface? In abstract terms, it is an interface that enables the user to reach his intended goal as fast, and easy, as possible. Usually, the user will have a general idea of what it is they wish to accomplish which becomes clearer the more they work towards it. The interface should provide the best tools, allowing the users to take actions, which move them closer to their perceived goal. When designing these tools it's important to understand that every design choice carries an associated benefit and cost. For example, when designing a new interface button, text will serve to minimize ambiguity, but has to be read to be understood; a graphical button is easy to recognize and aesthetically pleasing, but might be ambiguous in meaning at first. To complicate matters even further, this benefit and cost varies across many fields including cognitive psychology, human behavior, and graphic design. The actual benefit and costs are only determined in relation to alternative solutions. One potential solution might seem bad at first, but if it's better than all the other worse solutions, then it's still the best choice. The real list of factors that a design decision depends on can seem exponentially large. In practice, the problem is usually approached by following a number of design heuristics, previous experience, and a trial and error approach.

The choices that make up the final interface are very context dependent. An interface ideal for 2D graphics design might be completely unsuitable for 3D modeling and vice versa. But context dependency reaches far deeper than one might expect. The tools provided by the interface are custom made to fit the context, but the way the tools behave is also heavily context based. Specifically, the freedoms and constraints built into each tool. For example, the rectangular area selection tool from a graphics program often provides a constraint to allow the user to create a square instead of a rectangle. Symmetry holds a special place in graphics design and the interface restricts the user to only drawing symmetric objects if need be. The correct design of these freedoms and constraints go a long way to helping the user reach their intended goal while minimizing mistakes and avoiding unfeasible actions.

The perfect interface would allow every user to achieve their goal quickly and efficiently and leave every single one of them thoroughly satisfied. This is impossible. Not only because each person is different, and has different preferences as well as a different experience history, but also because each person usually wishes to accomplish something different given the interface provided. The last detail is particularly vexing when designing an interface since it crosses the border between interface and feature design. It is up to the designer to determine which features an interface will support and why. The design of an interface always starts with a need to fulfill a creative demand. Once the goal becomes clear, the interface design can commence. The design and inclusion of specific tools then rely on the requirements and constraints of the creative demand that is to be sated.

This process and associated topics is the focus of the remaining sections in this chapter. The details are presented as part of a usability study conducted on the PPP interface, explaining the relevant theory and its application.

4.2 Usability Study

Usability is the focus of interaction with a given product, and the ability of the user to achieve specified goals with effectiveness, efficiency and satisfaction [115]. While the first two aspects — effectiveness and efficiency — are immediately recognized as critical for successful interaction, the latter is less so. It is not hard to argue that if the user cannot efficiently achieve the specified goal with a high degree of effectiveness, then the product is unsuccessful. But why does satisfaction matter?

Satisfaction is key to motivating the user to continuously interact with a given product. It might seem counterintuitive at first, but there is no guarantee that satisfaction goes hand in hand with effectiveness and efficiency [115]. There is however, no doubt as to the results when it does. One of the best examples of a consumer product where satisfaction is at least as high as its effectiveness and efficiency, is arguably Apple's iPhone. While competing products offer comparable effectiveness and efficiency at lower prices, the user satisfaction is much higher with the iPhone [100]. In a sense it's almost better to have the unlikely combination of an inefficient and ineffective product that still results in satisfactory user experience, instead of an efficient and effective product that leaves users unsatisfied. All the efficiency and effectiveness in the world won't matter if your users refuse to use your product, and in the end, what comes first is the users willingness to use the product.

Effectiveness and efficiency are, of course, still of vital importance, and by extension; Usability as a whole. If there was ever any doubt to the importance of usability, one need look no further than studies involving medical interfaces. Jakob Nielsen [93] notes a medical study [64] within which 22 separate usability issues caused fatal mistakes to occur. Some of these issues were well documented in advance and theoretically did not require any specific testing to reveal.

As machines and computer continue their integration into daily use, the necessity and significance of proper usability testing becomes more and more important [115]. Usability testing is no longer just an added bonus, but can become the deciding factor for whether a given product fails in the task that it has been designed to do.

The PPP interface has been designed for the purpose of being integrated into a non-standardized communication process. The interface is motivated in part by the Danish pig meat industry and their increasing effort in optimizing product development efficiency, due to the ever competitive nature of the global meat market. The interface has been developed in collaboration with DMRI [28] and with assistance from Danish Crown [26].

The PPP interface was designed to accept input both from a typical mouse and keyboard combination, as well as a haptic feedback device. Each of these interface types will be analyzed in the comparative study. The intended user group is sales personnel with a pre-existing knowledge of pig product production standards. A cross section of sales personnel from Danish Crown volunteered to participate in the conducted study.

4.3 Overview

Each of the following Sections are divided into two segments. The first presenting relevant theory, and the second describing the application of said theory to the PPP interface to be examined in this thesis. The theory is presented more or less chronologically in the order it would be applied in a given usability study.

The following section details the critical aspects of planning and information gathering necessary, prior to any other steps in a usability study. Another early part of any usability study is selecting the optimal test participants, and is discussed in Section 4.5. Notable metrics often used in conjunction with user testing is described in Section 4.6, as well as potential pitfalls during application. Pilot studies conducted prior to the usability test and the resulting changes are detailed in Section 4.7.2. In Section 4.7 the actual usability test is described along with important details to consider in order to ensure clean data. Perspectives and conclusions are noted in the final section.

The DVD accompanying the thesis contains a compilation of the raw data gathered during both the pilot studies and the usability tests. Specifically, recorded video and audio, along with interaction logs generated by the PPP.

4.4 Planning and Information Gathering

Usability testing exists in many forms, from small informal in-house testing, to large formal external testing. But even with the smallest and most informal of testing scenarios, it is still useful to perform some basic planning, in order to maintain a good overview and control of the testing process. One of the main goals of planning ahead, is determining what to actually test for in the usability study. While certain issues will become apparent regardless of planning, a number of usability issues will remain elusive unless there is a proper focus achieved through proper planning. The optimal testing parameters, such as size, location, and number of participants, often relies on a number of factors, such as state of product development, availability of testing participants, availability of testing equipment, and so forth. Therefore, it is, in general, a good idea to plan the process.

The goals of the usability study will help define what type of metrics to choose in order to evaluate whether the test was a success or not. In fact, the entire method of evaluation might depend on what the end goal of the study turns out to be. Goals do not have to be defined down to the last detail, but the more

specific they are, the easier it is to choose the appropriate metric to ensure success.

In general, there are two types of usability studies [101, 115]:

- **Formative Study** — This type of study is conducted during the development stages of a product, when the design specifications are not final and features are still being added or modified. The goal of the study is to identify key usability issues and make recommendations to alleviate them. For example, what is keeping the users from achieving their goal while using the product, or, which aspects of the product are frustrating to use, or causes the user to make errors.

  It is not uncommon to perform multiple formative studies during development at regular intervals to ensure that precious resources are not being wasted on features the users find cumbersome or even superfluous.

- **Summative Study** — A summative study is usually performed at the end of product development, when the final design has been locked down and the product is ready to ship. This type of study is intended to identify whether or not the product lives up to the requirements established at the beginning of development. Competitive products are often included in the summative study as well as previous versions to better compare strong and weak aspects.

  Although this type of study is usually performed at the end of a product development cycle, the results should still be used for the purposes of improving the product. Evaluating the product without the intention for further improvement has little value.

Regardless of which type of study one commits to performing, it is vital to gather some preliminary information prior to any actual testing. Knowing what matters to the user, along with other relevant information, such as how often the user will be working with the product, is critical in shaping the usability study. With this knowledge in hand, it is possible to create real-world tasks that the user has to complete using the product. Having the user complete these tasks and applying various usability metrics will yield valuable data and reveal concrete usability issues related to them.

Apart from the significance of finding usability issues related directly to real-world tasks, they also help increase the users level of engagement in the testing scenario. If the tasks presented to the user are unrealistic and seems unlikely to ever happen in a real-world setting, then the participant is more likely to dismiss the test as a waste of time.

The easiest way of gathering this information is by communicating directly with the users of a similar product, or the potential users of the product being developed that has yet to exist on the market. If the interface being designed has been contracted by a company, it is likely that the actual users are the last in a long line of people you communicate with, due to the modern company hierarchy. Since the people at the top carry the responsibility of managing the development and integration a new product, they are also the first to discuss the matter. It is vital to note that what these people feel are crucial features and important aspects of the product design might vary greatly from what the actual users feel are crucial features and important aspects. I cannot overstate the importance of communicating directly with actual users of a similar product, if these people exist. If the product is an iterative design or intended to replace a current product, it is useful to observe the users in their natural environment using the product. Humans are not known for their precise memory, and if you only rely on interviews to determine how users interact with the current product, you are destined to only receive half the story. It is also important to emphasize the fact, that people will rarely be able to tell you outright what they actually need. It is up to you, as the expert, to see what they're not saying, and still make the proper decisions.

Another important part of any usability study is when to actually conduct it. There is no standardized approach, to determine which point in a products development cycle, is the optimal point in time, at which to conduct the study. My own personal rule of thumb is, the more often the better. Generally because developers always tend to test too little rather than too much. That being said, test results will yield more usable data, if there is a significant new amount of product to test. In other words, while it might be tempting to perform a study after every minor design change, it can become difficult to differentiate between noise and reliable feedback. The key is to strike a balance and test the product whenever a significant milestone has been reached during development. Between these milestones, impromptu testing by the developers themselves, is valuable for finding some of the most obvious usability issues, before subjecting actual users to the product in question.

Finally, when an actual date, time and plan has been set for the usability study, it is usually a good idea to perform a pilot study. A pilot study consists of executing the entire process of testing the software with a single volunteer from start to finish. It is highly recommended to find a participant close to the user group without any pre-existing knowledge of the system to be tested. In most cases, the pilot study will reveal a few unexpected events and issues with the testing procedure. If the discovered issues require a significant change to the test procedure itself and/or the system being tested, it is advisable to perform another pilot study to confirm that no more significant issues arise as a result.

4.4.1 PPP Planning

As a part of the planning phase of the usability study involving the PPP interface, industry experts and staff were interviewed in order to gain insight into the process of developing new meat products. Due to timing issues and the fragmented nature of developing new meat products, it was only possible to interview employees involved in the process, instead of observing the process directly. The interviews revealed the following main areas of interest for the PPP:

- **Product Reality** — A key factor for the success of the PPP is the realism with which it can depict the anatomy of the pig. The anatomy itself can almost be described as a cutting blueprint, which experts use as a guide to properly produce standardized products, despite biological variations in between pigs. A dynamic 3D pig carcass is only useful, if the same anatomical aspects of the pig can be recognized by experts, as on a static photo.
- **Factory Reminiscent Cuts** — The procedure of slaughtering a pig and producing consumer products involves a number of steps. In general, the first half of the procedure is primarily automated using machines, apart from the actual killing, which has to be done by a certified butcher. The second half is usually less automated and more done by hand, depending on how close to a consumer product, the final result is. The cuts committed by the PPP should correspond to real world applicable cuts, committed during the process of actual product development.
- **Ease of use** — Although this is practically a preferred requirement for any application, the relative ease of use is quite dependent on the intended target group, and their technical prowess. The target group in this case consists primarily of the sales staff which has received no formal training related specifically to the prototyping of meat products, and span a wide age gap from the mid-twenties to late fifties (24-58).

  It is also important to note that ease-of-use is the least easily observable trait of the three detailed. This is highly relevant, because the integration of the PPP is motivated by management, and not users. Lack of success in this area will not be deemed as critical by management as the other two areas. It is therefore the duty of the designer to be diligent enough to not let this fact negatively affect the outcome.

As previously mentioned, the PPP is unique as it does not replace an existing product, but instead is intended to be integrated to support a non-standardized

existing work flow. This leads to both advantages and disadvantages, from an interface design perspective. On one hand, users have not grown accustomed to a pre-existing interface. On the other hand, lack of standardization can lead to significant differences in user expectations/wishes. Getting to the bottom of these expectations and trying to fulfill them will provide the most useful feedback during usability testing.

Two significant design decisions during development of the PPP to fulfill these expectations was the use of volumetric data to represent the meat products realistically, and planar cuts as a good approximation of primary factory product cutting. A long slaughtering tradition has led to a standardized process where half a pig carcass is separated into three larger sections using planar cuts, as visualized in Fig. 4.1. These three parts are called "foreend", "middle", and "ham", from left to right. More specialized products are yielded using additional planar cuts, or precision cuts along the pigs specific anatomy, such as muscle membranes or bone structures.

Figure 4.1: Half a pig carcass, following the three division.

The separation of cut styles (planar vs. custom precision) are the result of the evolution of the production assembly line in slaughterhouses. The incentive to optimize production has automated as much of the production line as possible, leaving only the complex cuts to people at the very end.

Early in the design process, the possibility of simulating a real knife was considered as well as real 3D visualization. The Phantom Omni provides haptic feedback on three of its six axes, and modern stereoscopic technologies allow for more realistic 3D visualization than previously possible. Despite the recent advancements made within both these fields, initial prototyping did not yield confident results. However, the real deciding factor was the realization that the complexities and affordances provided by a realistic knife interface did not properly match the requirements of PPP interface. The simulation of an actual knife would impose real world restrictions on the interactions afforded by a vir-

tual world, which in turn would tend to inhibit the prototyping process, rather than accelerate it. A perspective on the next iteration of the tool, providing this custom interaction, is detailed in Chapter 5.

The Phantom Omni still offers advantages, not available in the more common input devices, such as the mouse and keyboard combination. The six axis interaction allows for the direct correlation between controller movement and virtual object movement. The traditional mouse interface is less ideal for the manipulation of 3D objects. Yet dispite the advantage of direct translational/rotational correlation, the Phantom Omni is arguably an exotic interface for most users. We therefore compare the two, to determine which interface is more usable.

Because a haptic controller is unfamiliar territory, not only to the potential users, but also the developers, a large amount of informal testing was conducted prior to planning the first formal external usability study. The informal testing with non-developers is useful to identify usability issues that have become obscured to developers due to overuse of the interface.

Some usability issues will only reveal themselves when the interface is tested with the intended user group, which happens to be the case with the formal usability study conducted in this project. The formal usability study is best described as a formative study, since it involves the comparison of two potential interfaces, leading to comparisons revealing the pros and cons of using either interface. In addition to the requirements established during interviews with the experts and staff from the meat industry, some more specific usability goals were defined as follows:

- Was the user able to re-create existing meat product using both interfaces?
- Which of the two user interfaces did the user prefer? and why?
- What advantages/disadvantages did one interface carry over another, for the user?
- Which, if any, significant usability issues prevented the user from completing the tasks provided?
- What aspects of the interfaces worked well, and which were frustrating?
- What were the most frequent errors made by the user during the interaction?
- How was the users perception of usability either interface?

Of all the goals listed above, one stands out in particular. The first goal is the only one that cannot be evaluated properly, without additional expert knowledge. Early in the process we realized that the actual end users do not posses the expert knowledge required to recreate existing products with a high degree of accuracy. Regardless of this supposed lack of expertise, we can still only gain from having a ground truth with which to evaluate the results objectively.

This expert knowledge was collected with the help of factory manager Jesper Frandsen, at an abattoir owned by Danish Crown [26]. With his aid, a total of five standardized products were created and recorded for future comparison, with the PPP. Although Jesper was not the intended target group for the PPP, his help and interaction proved exceedingly valuable, as described in Section 4.7.3.

The metrics to evaluate how well the remaining goals were satisfied are described in Section 4.6

The actual structure of the formative usability study is described in Section 4.7.

4.5 Test participants

Any proper usability study will need the support of outside test participants to provide a perspective not affected by the product development experience. It is quite common for developers to be blinded to flaws and failures related to the products they are working on. Rubin and Chisnell [101] note that there is no better evidence to convince a skeptical developer of usability problems, than that of a struggling participant.

Although it is always preferable to test a product with its intended target user group, there are instances where this is not possible. Usually the main cause for this, is that the intended users are not available in abundance, and/or the available users have better things to do. Even so, users outside of the specifically targeted user group are still much more useful than anyone directly connected to the product. By enlisting the aid of users who are as close to the target user group as possible, the likelihood is increased that relevant usability issues are discovered during testing.

If the product is being developed in a collaboration with a company intended for use by its employees, it is likely that you will have direct access to the intended target group. In this case, it is important to be wary of only receiving the "best" employees for your study. Participation in a usability study will sometimes be

viewed as a reward by company management and will be reserved for the employees they feel perform the best. Unfortunately, only testing the product with the "best and brightest" will be detrimental to the results yielded by the study. It is crucial that the product is intuitive and easy to use for everybody, especially users who may not be overly familiar with any similar product. Consequently, it is important to stress that the users involved in the testing represent the entire spectrum in both age, gender, competence, experience and any other relevant categories.

The number of test participants will determine the reliability of the results and varies depending on whether a formative or summary usability study is being conducted. For a formative study, 6-8 test participants are usually sufficient for yielding reliable results [115]. Tullis and Albert [115] suggest at least 50-100 participants when conducting a summary study in order to keep variance low and generalize the findings for a broader population. Rubin and Chisnell [101] note that the number of participants is also indirectly determined by the variation in your user target group. The more widespread a demographic, the more testers needed to reliably determine usability issues pertaining to each category.

The PPP is intended to primarily be used by sales staff during meetings with current and prospective clients, while defining specifications for new meat products. Meetings with the staff and management has revealed that the sales personnel is most easily categorized by age. A younger group aged 24-37 consists of newer sales personnel with little to no hands-on knowledge of meat handling, and an older group aged 45-58 with a background more tightly connected to the abattoir. Both groups are technically proficient with computers and use the keyboard and mouse on a daily basis. None of them had had any previous experience with the Phantom Omni or similar 6 degrees of freedom interface, and neither had any of them have undergone formal training sales training.

A total of 8 volunteers participated in the formative usability study.

4.6 Performance metrics

In this context, metrics refer to a standard of measurment. They enable measuring something objectively. The advantage of applying standardized metrics, is that two separate individuals applying the same metric produce comparable results. All fields of research usually have some kind of applicable metric, and the usability field is no different. Metrics within this field reveal something about the user experience, such as how easily the user achieved the specified task, or how satisfied the user was after having completed the task.

General metrics commonly applied for various purposes during usability studies include the following:

- **Task Success** — Simple binary metric for evaluating if the user is able to solve the specified tasks using the product. In many cases, the actual result is not nearly as useful as the observations made along side the user as they complete the tasks.
- **Time to Completion** — A timed metric detailing the amount of time a user needs in order to complete a specific task. This metric is a good indicator of overall efficiency.
- **Errors** — Certain interfaces might allow the user to commit actions that nonsensical or essentially a waste of time. Not necessarily due to poor design, but simply as a result of the inherent flexibility of the interface. It may be hard to prevent the user from actually performing these useless actions, but detecting them is easy. By noting how many actions the user carries out that has no real effect, one can determine whether or not it might be necessary to change a fundamental part of the interface design.
- **Questionnaires** — Apart from observing the user directly, this is the most useful metric to assess their subjective experience of using the product. A number of different standardized questionnaires exist, such as "System Usability Scale" (SUS) [16], "Software Usability Measurement Inventory" (SUMI) [60], "Questionnaire for User Interaction Satisfaction" (QUIS) [50], and "Usefulness, Satisfaction, and Ease of use" (USE) [81]. All of these standardized questionnaires are the next best thing to actually measuring the users response directly during interaction. While the results are highly subjective, they are also provide useful insight into how the product is perceived by the user.

Which metrics to apply during a given usability study depends on a number of factors. The primary deciding factor being the goals set forth prior to the actual study, since the metrics are crucial for providing the data that complete the individual goals. Secondary factors include the amount of time available to conduct the study, number of participants available, as well as the budget determining what type of equipment can be used during the study itself. Since the end goals of each usability study will vary significantly, it is impossible to tell in advance what metrics are going to be useful prior to actually planning the study without knowing the specifics.

Not knowing precisely which metrics to apply could lead to the tempting scenario of applying as many metrics as possible. Tullis and Albert said it best

when they stated "There are only so many aspects of the user experience you can quantify at any one time" [115] (p.296). It is much more fruitful to thoroughly examine and commit to the most applicable metrics in a given situation. In the same sense, it is vital not to rely too heavily on a single metric. It is unlikely to be able to represent the entire user experience, and will lead to missing vital data.

According to Tullis and Albert [115], when comparing products (or in this case interfaces), the three recommended metrics are task success measures, efficiency evaluation, (such as time to completion,) and finally a self-reported satisfaction metric. Therefore, it is not especially surprising that the metrics chosen to evaluate the PPPs interfaces are as follows:

- **Task Success** — Since the PPP is arguably at a prototype stage, it is important to ensure that the tasks it was designed to do are actually possible. Since the overall goal of the PPP is to aid the communication process involved in designing a new meat product, a number of tasks will involve designing a specified product given half a pig carcass.
- **Time to Completion** — Ease of use is important since the PPP is intended to be used in a sales scenario. A frustrated user could easy lead to an annoyed customer, so the application must not hinder the user in trying to achieve the intended goal. How long it takes for the user to complete the aforementioned tasks is a good indicator of which interface is more easy to use.
- **Questionnaires** — Having completed the specified tasks, the user will be asked to fill out a custom questionnaire, based on the SUS questionnaire [16]. It is important to be wary of overly positive feedback. Having never used a haptic feedback controller, it is not uncommon for users to be positively biased towards new and intriguing experiences.

In addition to the chosen metrics, we are also interested in revealing as many usability issues as possible during testing. Therefore, we will also ask the participants to think aloud during the test to identify issues not automatically revealed through metrics.

4.7 Formative Usability Study

Once the planning has been completed, the target user group found, and the metrics decided upon, it is time to prepare the system so that it is test-ready.

If the metrics chosen to evaluate the study with are all clearly observable by the testing staff, then no modifications are necessary. In most cases however, the system will need to have a number of additional functions added, such as logging or resetting.

But, regardless of the changes made to the system, it is always wise to perform a pilot study before conducting the actual usability test. It's quite common for the pilot study to reveal a number of unexpected issues. If the number of issues found are significant, an additional pilot study should be conducted to ensure that they have been handled properly.

Once the system is ready for testing and the entire process has been planned out from greeting the participant to thanking them for their time and sending them on their way, it's important to spend some time considering where to conduct the usability study. The most suitable answer, regardless of the type of study being conducted, is wherever the user is likely to be most comfortable. The user should feel at ease and not be distracted or bothered by unfamiliar elements. Arguably, the most relaxed and undistracted state the user can be in, is being unaware of any testing. But apart from the difficulty of having users perform tasks without them knowing they are being observed, it is a moral and legal gray area. The proper approach is therefore picking a scenario that will help keep irregularities and distractions to a minimum during testing, for both the benefit of the user, as well as the resulting data.

4.7.1 PPP Testing procedure

Since the usability study for the PPP involves multiple interfaces, the test is reminiscent of a comparative study. In such a study, a certain bias is likely to develop for the interface the user is presented with first. Additionally, performing the same task with similar interfaces will negatively impact the end result of the study [115]. However, while the tasks performed during our usability study share similarities, the two interfaces differ significantly in their use. The only real similarity between the interfaces is the visual presentation. The learning effect from using either of the interfaces will therefore be minor. Despite this fact, we still counter balance for good measure by presenting half of the participants with one interface first, and the other half the second interface.

As noted in Section 4.6, a number of metrics were selected that require the PPP to output data for later interpretation. Modifications were made to the application to allow for recording user interaction, as well as supporting an established testing plan. As none of the participants will have interacted with either of the interfaces before, it is necessary to start each test with a brief

tutorial. In order to further practice navigating the virtual scene using either interface, a set of simple navigational challenges were constructed, which the users must complete. Then the users are tasked with recreating a set of five standardized pig products. Finally, they are given the questionnaire to fill out, after which the testing for that interface is complete, and the process is repeated for the alternate interface.

Figure 4.2: The meeting room within which the usability test was performed.

The entire test took place in a meeting room, pictured in Fig. 4.2, in the building housing the sales staff volunteering for the usability test. Each of the test procedure segments are described in further detail in the subsections below.

4.7.1.1 Interface Tutorial

The ability of users to figure out a given system without any instructions should not be underestimated, provided that each action has a clear and distinct reaction. However, even with a clear pattern of cause and effect, it is always possible for a user to misinterpret the causal relation, which can lead to disastrous consequences if not caught and handled early.

It is wise not to allow for any opportunity of ambiguous interpretation. There-

fore, inform the users from the very beginning about how to interact with a given system and provide time to ask questions and practice.

Another ideal method of teaching users is through tutorials. The art of creating tutorials has undoubtedly been advanced by the influx of games into main stream, as almost every single major published game includes a tutorial in one form or another. Some tutorials are restricted to just a simple text prompt informing the users of which controls correspond to which actions. The most advanced tutorials display additional visuals to clarify complex mechanics and even impede the users progress until they have successfully demonstrated proper use. The best tutorial to implement depends on a number of factors; the end user, the context of the tutorial, the severity of the idea being conveyed. But regardless of any of these factors, an interactive tutorial will always be more engaging for the user than simple text prompts on the screen.

In the case of the PPP, an automated interactive tutorial would also be ideal. Each participant would experience exactly the same instructions and interactively tested for understanding of the simplest of concepts. We chose to compromise on the tutorial design for two reasons:

- **Lack of time** — The complexity of creating an automated tutorial for a prototype interface increases as the functionality of the interface grows. Although both the interfaces created for the PPP are simple in nature, an automated process to ensure the users understanding is not. In fact, ensuring that a tutorial is complete and understandable by the majority of users will often require its own usability test.
- **Scope of application** — The prototype application is almost guaranteed to change as a result of the usability test. If not, then the test would have to indicate that the users are 100% satisfied. Even the most experienced interface developer cannot hope to achieve such success without having tested the interface even once. Investing a large amount of time for the creation of a one-time use tutorial is rather inefficient, especially when there are perfectly viable compromises.

The end result is a combination of pre-recorded video sequences with live narration, followed by an interactive practice session aided by testing personnel. This allows us to ensure the that participant grasps the basic functionality of each interface during a practice session where they are free to experiment with the interface.

As previously mentioned, two interfaces for the PPP are compared based on the Phantom Omni and mouse, respectively. Each of these interfaces have one

functionality mapped to a common key on the keyboard. Switching between the two different types of rendering, visualized in Fig. 4.3, is accomplished by pressing the `space bar` on the keyboard.

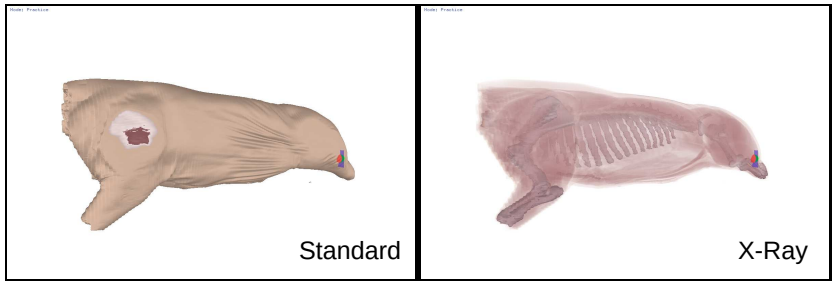

Figure 4.3: Screenshots of the two different rendering options available to the user. On the left, the standard surface rendering mode. On the right, the x-ray-like render mode allowing the users to see the bone structure of the pig carcass more clearly.

The mouse interface allows the user to rotate, zoom and pan the pig carcass by using the left, middle, and right mouse button respectively. Due to the limited degrees of freedom provided by the mouse, rotating and panning is limited primarily to the horizontal and vertical viewing axis. Rotation is similar to the virtual sphere interface as described by Chen et al. [22]. The rotation applied on to the pig carcass is best described by imagining a virtual sphere in the center of the screen. The further away from the center of the view port, the users drags the cursor to affect a rotation, the more the third axis (depth of view) affects the rotation. If the mouse is moving fast enough when the user releases the left mouse button, the pig carcass will continuously rotate in the same direction until any mouse button is pressed.

While using the mouse interface, the user creates a cutting plane by placing the cursor on top of the pig carcass and pressing the `enter` key on the keyboard. This creates a cutting plane centered in the cursor removing the left part of the pig (on the red side of the cursor). Just as with the pig carcass, the cutting planes are rotated and moved with the left and right mouse button respectively. Each plane is interacted with individually, i.e. it's only possible to manipulate a single plane at a time. The user can rotate the plane along an imagined centered axis by grabbing an edge. Figure 4.4 illustrates the user grabbing the vertical edge and rotating the plane along an imagined vertical axis down the middle of the plane. Alternatively, the user is free to grab any corner of the cutting plane and allow the user rotate the plane along an imagined diagonal axis, spanning the two neighboring corners to the one the user has selected. Each of these rotational scenarios are also visualized in Fig. 4.5 along with an imaginary line that will always intersect the plane. Finally, the user is also able to reposition the plane along its normal by using the right mouse button.

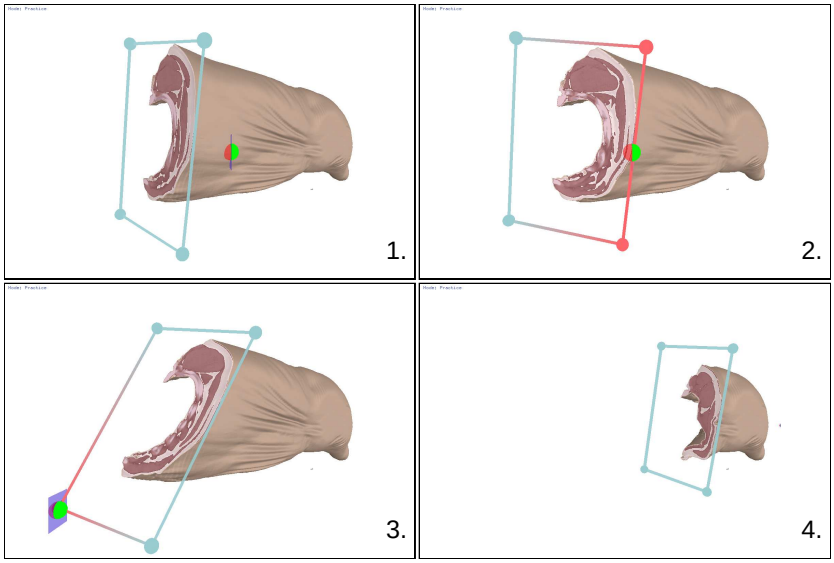

Figure 4.4: Screenshots of mouse interaction with the cutting planes. 1. A single cutting plane intersecting the pig carcass. 2. The mouse cursor interacting with the edge of the cutting plane. 3. The mouse cursor interacting with the corner of the cutting plane. 4. The plane being repositioned along the pig carcass.

The rotation and translation of the planes is simpler in comparison to the rotation and translation of the pig. This is due to the limitation of ways the user can interact with the plane while still having it centered around the pig. We restrict the rotations and translations such that they only allow results that would still have the cutting plane intersect with the pig.

The one exception being, when the user wishes to delete a plane. To delete a cutting plane, the user simply moves the cutting plane along its normal until it no longer intersects the pig carcass.

When using the Phantom Omni, which provides six degrees of freedom, we found the most intuitive interaction style was to simply allow the rotations and translations of the pen to be directly translated to the pig carcass and associated cutting planes. Preliminary testing also indicated that it was easiest to assign all interaction with the pig to one button, and all interaction with the planes to the other. The button nearest and furthest from the tip of the pen on the Phantom Omni, pictured in Fig. 4.6, will be referred to as the front and back button respectively.

We assigned all of the plane interaction to the front button on the Phantom

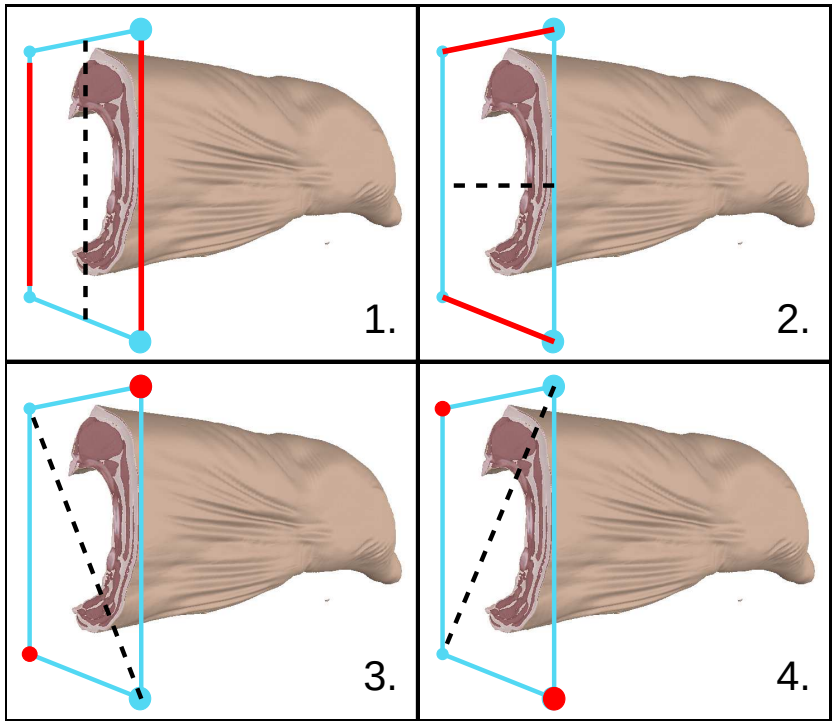

Figure 4.5: Four rotational scenarios along with an imaginary line that will always intersect the plane depending on the rotational scenario. The highlighted red parts are "grabbed" by the user, while the dotted line is the imaginary constantly intersected line. 1. Grabbing the vertical edges. 2. Grabbing the horizontal edges. 3. Grabbing either of two opposed corners. 4. Grabbing the alternate two corners.

Omni. Creating a plane is done by simply clicking the front button without hovering the cursor near an existing plane. If the button is held while near an existing plane, the user is free to rotate and reposition the aforementioned plane as long as the button is held. Holding down the back button allows the user to rotate and reposition the pig carcass.

To delete a cutting plane, the user simply grabs it and drags it away from the pig carcass.

The haptic capabilities of the Phantom Omni was also put to use, providing feedback if the user touched the pig and having a slight magnetic pull towards nearby cutting planes.

Figure 4.6: The Phantom Omni © Copyright Sensable Technologies, Inc. Image user with permission by Sensable Technologies, Inc.

4.7.1.2 Navigational Challenges

To engage and quantitatively measure the participants using the interface, a series of navigational challenges were created. The user was asked to align a set of navigational frames with the view window, visualized in Fig. 4.7. A total of five navigational challenges were created. The location and orientation of the frames differ slightly depending on which interface the user is engaged in testing. Differences in the frames were a necessity as the mouse interface was somewhat more restricted in its user freedom, given the inherit lack of degrees of freedom. The difference in navigational frames also served to reduce the learning effect in between tests.

The proper alignment of the navigational frame was automatically detected by the application at which point the next remaining frame would be presented, and the time used logged for future use.

4.7.1.3 Product Creation

In this phase of the usability test, the user is tasked with recreating a total of five standardized pig products. The participants have pre-existing knowledge

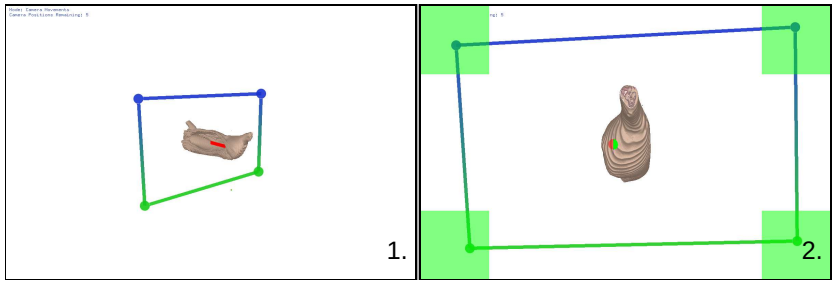

Figure 4.7: Screenshots of the navigational frames the user is presented with, and has to align to the corners of the view window to proceed. The two green corners must be aligned to the bottom corners and the blue to the top corners. 1. On the left is a screenshot of an, as of yet, unaligned navigational frame. 2. When any of the corners are properly aligned, the system provides feedback by displaying a green square in the respective corner.

regarding all of the products they are asked to recreate. To ensure that there is no confusion or ambiguity regarding the products they are tasked with creating, every participant was provided with images of each of the products in question. After all, the test is intended to identify usability issues, not evaluate how well the users can recreate products from memory.

Real-time feedback on the user's accuracy in recreating the intended products was considered for this phase. However, the actual goal of the task was not to recreate as accurate a product as possible, but rather determine the ease with which the user could create an approximation of it, as well as determine which usability issues arose during interaction. Any feedback indicating how much the users recreation of a given product differs from the expert version would undoubtedly cause the user to focus on improving their accuracy at the cost of performing the task quickly. This would skew the timed measurements to reflect how persistent each user was in achieving as close a reproduction as possible to the expert product. Therefore, we decided to leave the evaluation of the finished product to the users themselves.

The user was instructed to indicate when he/she felt the product was complete. At that point, the created cutting planes are recorded, along with the time used to create the product.

4.7.1.4 Questionnaire

A custom questionnaire including the SUS questionnaire, as originally published [16], was presented to the user after they had completed the previous phases with a single interface. Once both interfaces had been tested, we presented the participant with an additional binary choice to determine their preference for either interface, along with asking them to justify their choice.

4.7.2 Pilot Study

As previously mentioned pilot studies are vital to determining problems with the testing procedure. A complete pilot study was conducted with a colleague who had never used either of the interfaces before. The following issues were discovered as a result:

- **Questionnaire** — One of the questions from the SUS questionnaire was missing from our printed version.
- **Missing navigational challenge** — A navigational frame for one of the interfaces was missing.
- **Graphics glitch in mouse interface** — When the cursor neared the edges of the screen in the mouse interface, a graphical error would occur where the cursor would increase in size and fill the entire screen. A relatively insignificant problem, which was simple to fix.
- **Navigational challenge difficulty** — The navigational challenges posed to the user were determined to be much too difficult. Specifically, the corners with which each navigational frame had to be aligned caused the user a lot of difficulty. This is a classic example of the developer becoming too acquainted with the software, so as to lose touch with the inherent difficulty in attempting something for the first time, without prior practice.
- **Lack of visual cues to aid in cutting** — Initially, the cursor did not indicate which side each newly introduced cutting plane would remove pig carcass tissue. While not specifically an issue during testing, it became clear that this type of visual feedback was clearly beneficial for the user, and therefore implemented.
- **Haptic feedback vibrations** — The magnetic pull from nearby cutting planes would occasionally cause the Phantom Omni to start vibrating due to multiple contradicting forces. The pull generated by the cutting planes was reduced to prevent this from happening.

- **Subtle visual interface** — The colors used to highlight the different portions of the selected cutting planes were intensified based on user feedback.

Since the list of changes made to the PPP was significant, a second pilot test was performed with yet another colleague to identify any outstanding issues. The following final issues were addressed as a result of the second study:

- **Right to data collection** — The agreement, to be signed by participants, allowing us to collect and analyze usage data anonymously, was clarified and made more precise.
- **Redundant haptic interaction** — The haptic feedback caused by interacting with the pig carcass during navigational challenges was deemed redundant and disabled. The user has no need to 'touch' the pig while trying to align the frames.
- **Rotation speed increase** — The user complained that the rotating of the pig carcass was not responsive enough. The rotational speed using the mouse was doubled as a result.
- **Challenge skipping** — Although the navigational frames were much easier to align to the screen, it was deemed important to implement the possibility to skip individual phases should the need arise. Although not collecting this type of data would be a loss, it was important to prioritize the creation of products as the main focus of the testing. It is conceivable that a user might be able to properly prototype products, yet fail at aligning the navigational frames.

The issues identified by the second pilot study were minor in nature and caused minor changes to the interface after being corrected. A third pilot study was therefore deemed unnecessary.

4.7.3 Expert Product Creation

Expert knowledge was gathered from a trained butcher, Jesper Frandsen from Danish Crown [26], for the purposes of recreating and collecting cutting data of five standardized products. Since the primary goals were to collect expert knowledge and assess product reproduction, an expert user cooperated with the trained professional butcher in recreating the standardized products. The expert user interacted with the PPP interface guided by the professional who provided

instructions as to where to apply the cuts to the pig carcass. Photographs of the five products along side the virtual reproductions can be seen in Fig. 4.8.

Figure 4.8: The five products shown alongside their virtual counter parts. The virtual products shown in the bottom are created using the expert cuts. The most striking visual differences are caused by a lack of physical simulation, most notable in product 1801, where the cut flesh does not even out at the end like in the real world. The other notable difference is skin appearance, caused by interpolation, most notable in product 1602 and 1801.

The volumetric representation of the pig carcass easily allowed for the trained professional to recognize both bones and muscle groups, and made significant use of both the standard and x-ray-like rendering of the pig carcass. The low quality rendering mode did not inhibit the butcher from recognizing significant anatomy used as a guide to apply the cuts, which allowed the cuts to be placed quickly and precisely aided by the expert user. The previously mentioned visual artifact of malplaced skin color also did not negatively affect performance.

It is possible to provide a rough general measurement of the accuracy of the applied expert cuts, by examining the first three products visualized in Fig. 4.8. These three products together make up the entirety of the pig carcass from which they are each individually cut. Out of the `4261045` voxels comprising the pig carcass, the first three expertly cut products together make up `4262697`, yielding a difference of `1652` voxels, less than `0,1` %. It is, of course, possible that the first product includes the whole pig, and the other two cut it away completely, yielding the best accuraccy, but as Fig. 4.8 shows, this is not the case.

4.8 Results

There are numerous ways and methods with which to interpret results gathered from a usability test. Each of those methods often have their own strengths and weaknesses, with many being ideal in some, but never all, situations. The usefulness of the methods are also highly dependent on the data provided. It is beyond the scope of this thesis to cover, even just the basics of, all these different methods. Instead, generalized best practices are noted upon as well as potential pitfalls during interpretation, before providing the specifics behind the data and associated analysis performed on the PPP.

As with most kinds of data, it usually helps to view them from as many perspectives as possible in order to recognize patterns or "hot spots", which call for further attention. A modern spreadsheet enables easy visualizations of the data using a large variety of display styles. However, this multitude of options is only useful if care is taken to separate the truly important from the noise. Especially when communicating any significant findings when presenting the data.

Having compiled the findings into a compact, easy to understand and display, visualization, it is important to report the confidence interval [109] associated with them. Confidence intervals reveal how likely it is that your findings are also valid for the remaining population that did not participate in the usability test. Without the confidence interval is hard to say anything with any certainty regarding the collected data. Section 4.8.1 presents all of the results collected during the usability study.

4.8.1 PPP Results

The quantitative data gathered during the usability test, including a short evaluation, is provided in the following subsection. It is important to emphasize the likelihood of a bias in any results from a usability test comparing as common an interface as the keyboard and mouse with any other interface. Even though it is a priority to minimize bias whenever possible, it is also important to remember that the large majority of users in the case of the PPP are likely to have pre-existing experience with the mouse. So while the results are not an unbiased evaluation from a person with no pre-existing knowledge, the results do resemble the real life scenario of a user already familiar with the mouse and keyboard.

In the case of the PPP, none of the participants had interacted with either of the specific interfaces before, so despite the intimate knowledge of the affor-

dances provided by the mouse and keyboard, they are still novices when using the mouse-based interface of the PPP. However, there is undoubtedly a higher familiarity with that particular input device and it might result in the user having an easier time learning to use the interface.

Each participants age, gender, first interface usage, as well as their final personal preference is listed in Table 4.1.

Participant	Age	Gender	First Use	Preference
1	32	Male	Mouse	P. Omni
2	49	Male	Mouse	P. Omni
3	53	Male	P. Omni	Mouse
4	28	Male	P. Omni	Mouse
5	45	Female	Mouse	Mouse
6	37	Male	Mouse	P. Omni
7	24	Male	P. Omni	P. Omni
8	58	Male	P. Omni	P. Omni

Table 4.1: The total number of participants in the usability study, along with their age, gender, which interface they were presented with first, and which they expressed a preference for.

On a single occasion, during the usability test for the second user, the PPP application crashed. Unfortunately, this meant that the participant could not provide us with a complete set of data. This discrepancy is reflected in the confidence intervals calculated using the data, by the lower sample base. Coincidentally, the same user was also unable to complete the navigational challenges using the Phantom Omni within the allotted time. Again, the confidence interval reflects this discrepancy by calculating results with a lower sample size.

4.8.1.1 Quantitative Data

This section presents two different types of summarized quantitative data: per user and per task. The data is summarized using varying statistical methods, as elaborated below:

- **Per User** — We summarize the average times per user, using the mean and standard deviation. These two measurements give a clear picture of the users overall performance given the set of differing tasks. The measurments are not particularly important for the evaluation of the interfaces on their own, but are presented for the sake of full disclosure.

- **Per Task** — Per task summaries are more interesting, since they are multiple measurements of users trying to accomplish an identical goal. Sauro and Lewis [105] have shown that the geometric mean is a better estimate of the middle task-time in usability studies, than the arithmetic mean or median, for a small sample size. Therefore, we apply the geometric mean and calculate a 95% confidence interval using the geometric standard deviation.

The mean time required for each user to complete the tasks presented during the navigation phase are visualized in Fig. 4.9. The data shows that there is a clear difference in skill between participants, and the results from the two interfaces seem to be slightly correlated.

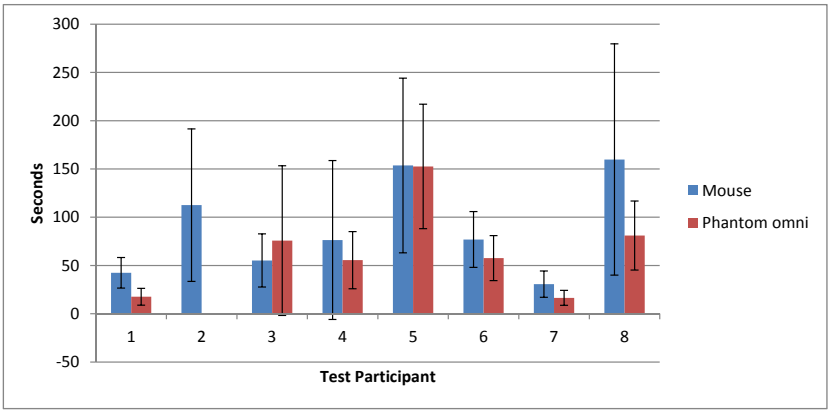

Figure 4.9: The mean time taken for each participant to complete the five navigational challenges, along with the standard deviation for each of the two devices. As previously noted, the second user could not complete the navigational challenges within the alotted time and is therefore left unreported.

Every person is different, so it really does not come as a surprise that the competence with the interfaces differs between persons. This type of data would not normally be included in a summary since it does nothing to prove or disprove any of the usability goals, except give a vague idea of how long on average it takes these individual people to complete the navigational tasks.

Figure 4.10 visualizes the same data, summarized per navigational task.

The mean navigational time is an approximate estimate for how much time we can expect users to require, to solve a given task, using either the mouse or the Phantom Omni as input device. Since our sample size is relatively small, the confidence interval is quite large as Fig. 4.10 clearly shows. The data has a

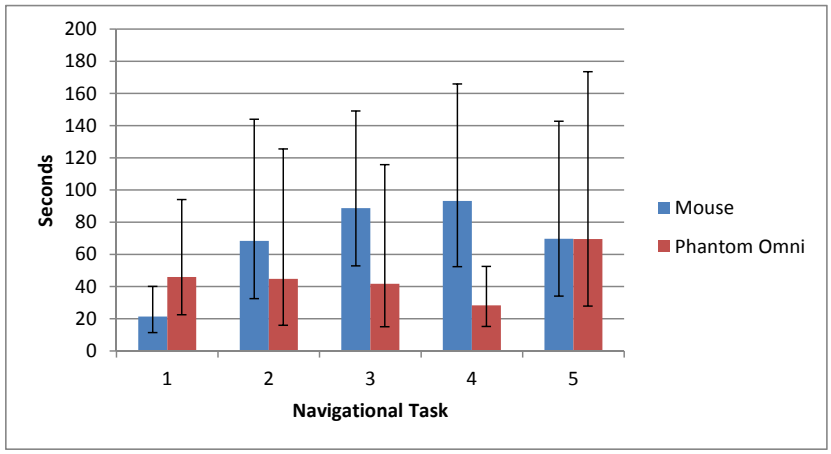

Figure 4.10: The geometric mean time taken by users to solve the individual navigation tasks using the mouse and the Phantom Omni. The error bars represent the 95% confidence interval using the geometric standard deviation based on the students t distribution. Because the standard deviation is calculated on a logarithmic scale, the upper and lower confidence interval bounds are not equally distant from the geometric mean.

slight lean towards the Phantom Omni in overall fastest task completion, but it is too small to say anything with certainty. As previously noted, there is a difference in between the tasks presented to the users during the navigation phase, as each of the interfaces navigate the virtual space differently. Thus, a direct comparison between measured task times from both interfaces can be helpful, but should definitely not stand alone as an assertion of a specific interpreted result.

The task times for each user to create all of the five products are a more solid basis for comparison. At that point in the usability test, each user has been given an introduction to the interface, practiced, as well as solved a series of challenges designed to familiarize the user with the interface.

The mean time each user took to complete all of the products is visualized in Fig. 4.11. Just like in the case of the navigational tasks, the individual mean time per user is not particularly useful, except for showing that each participant is not equally capable with both interfaces.

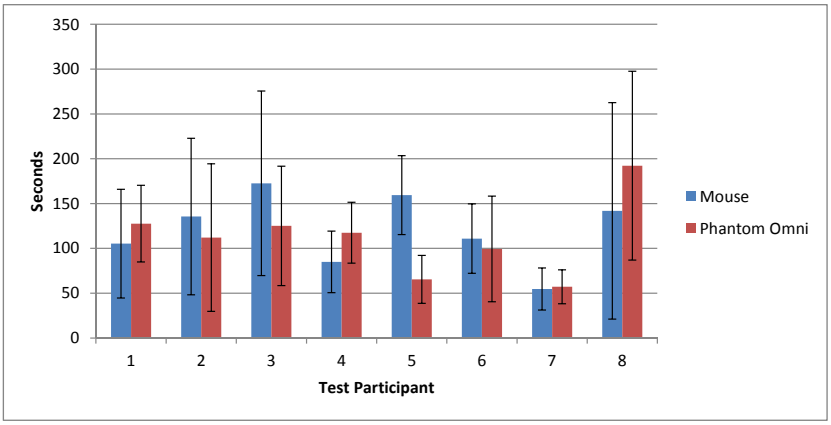

Figure 4.11: The average time taken for each participant to create the five meat products, along with the standard deviation for each of the two devices.

Figure 4.12 visualizes the geometric mean time for each of the five created products. The last two products are undoubtedly the most complex to create, which the measured time clearly reflects. The results do not clearly favor one interface over the other. Even though it's relatively clear visually, given the upper and lower confidence intervals, it is still advantageous to perform a more thorough analysis to determine whether or not the results quantitatively lean one way or another.

We perform a 3-way analysis of variance (ANOVA) [86] (mixed effect) with pairwise interactions, where the interfaces and product creation tests are fixed effects and users are considered a random effect. As previously mentioned, we intentionally do not compare the navigational tests as they differ by design, and serve as part of the learning introduction.

The results, displayed in Table 4.2, show no significant effect from the interface alone. Unsurprisingly, the analysis indicates that certain products are significantly easier/faster to create than others. It is also clear that there is a significant difference in performance with the interfaces per specific user. However, these findings are not relevant for the comparative study between the two interfaces.

The findings reinforce the interpretation that, while there is a clear variance associated with what type of task is being completed or the interface being used for that particular task, there is no clear discernible performance difference between the interfaces overall.

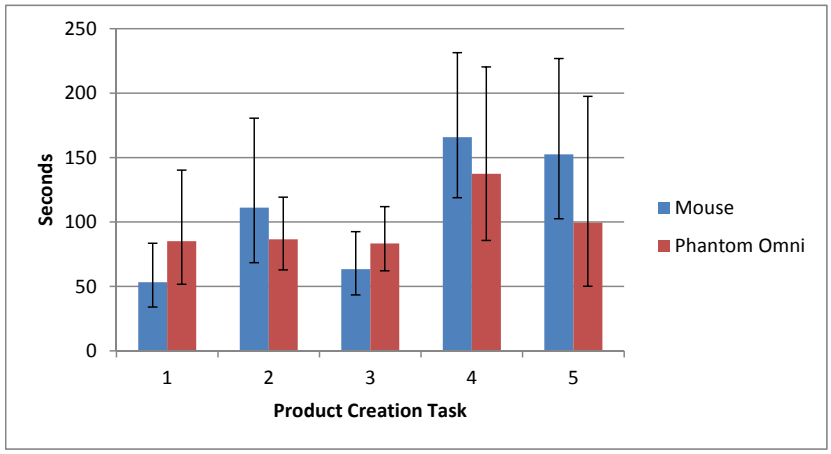

Figure 4.12: The geometric mean time taken by users to create each of the five meat products. The error bars represent the 95% confidence interval using the geometric standard deviation based on a students t distribution. Because the standard deviation is calculated on a logarithmic scale, the upper and lower confidence interval bounds are not equally distant from the geometric mean.

Source	Sum Sq.	d.f.	Mean Sq.	F	Prob>F
User	5.3925	6	0.89876	1.49	0.3308
Interface	0.0001	1	0.00006	0	0.9928
Test no.	5.5283	4	1.38208	8.56	0.0002
User*Interface	3.7328	6	0.62214	3.48	0.0129
User*Test no.	3.8732	24	0.16138	0.9	0.599
Interface*Test no.	4.2952	4	0.56971	3.18	0.0312
Error	4.2952	24	0.17897	0	0
Total	25.1009	69			

Table 4.2: Results from the analysis of variance with pairwise interaction, where interfaces and products are fixed, and with users considered a random effect.

The scores collected by the SUS questionnaire are shown in Table 4.3 along with a calculated upper and lower confidence level. The scores are, again, too similar to declare one interface more usable than the other. However, it is possible to interpret the general perception of both of the interfaces. Bangor et al. [9, 10] have analyzed a large body of work using the SUS, and established an overall grading scale for evaluating scores. An average score around 55 places both these interfaces in the "average" category. A clear indication that neither is perfect in its current form.

Interface	Mouse	Phantom Omni
Mean SUS Score	55,3125	56,875
Lower 95% Confidence	46,846	46,898
Upper 95% Confidence	63,779	66,852

Table 4.3: The mean SUS score for each interface along with upper and lower 95% confidence.

4.8.1.2 Qualitative Evaluation

Having the PPP interfaces tested by eight different people provided valuable insight into where problems occurred, and what might be ideal changes to the interface. Below is a list of the significant observations and interpretations of the usability test:

- **User fatigue** — A single case of user fatigue was noted during testing with the Phantom Omni interface. While an isolated case, we believe the significance of this should not be overlooked. For the Phantom Omni to be optimally incorporated into any daily use, it will require its users to acclimate themselves to becoming fairly agile with their wrists. The age of the user and by extension the expected health of the users various muscle groups have a significant impact in regards to the Phantom Omni, in our opinion. Zhai et al. [133] also reported a case of user fatigue in their study of muscle groups affecting performance.
- **Plane Selection** — On a few occasions, test participants experienced difficulty in selecting a single isolated plane. We considered solutions for each of the interfaces to overcome this issue. In the case of the mouse interface, the thickness of the cutting plane frames could be adjusted to fit the number of nearby planes to allow for easier selection when they are isolated. In the case of the Phantom Omni interface, a direct line in between the cursor and the nearest plane might help mitigate issues with perceived depth.
- **Control Confusion** — When using both the mouse as well as the Phantom Omni, users would occasionally mistake the function of one of the buttons with that of another. Additional visual and audial cues might help users better remember which buttons are assigned which functions as well as longer, more tutorialized, introductions.
- **Navigation Plane** — Three users initially tried to use the cursor to grab the navigational planes during the navigational testing phase. Making them less similar to that of the cutting planes would likely resolve this misunderstanding.

- **Phantom Omni Continuous Button Holding** — During preliminary testing we experienced occasional problems in holding down the Phantom Omnis buttons continuously. We also noticed similar issues during both pilot tests as well as during the usability study. We believe this to primarily be the result of low quality buttons on the Phantom Omni.

- **Mouse Rotation Stopping** — The mouse allows for the user to continuously rotate the pig carcass, if the cursor is not static while letting go of the left mouse button. This feature would occasionally cause the user to unintentionally rotate the pig carcass. Since few participants actually used the feature, it would either be best removed, or made more less sensitive to the users movements when letting the mouse button go.

- **Mouse Rotational Confusion** — On a number of occasions, users would verbalize having trouble figuring out how to rotate the pig carcass using the mouse. A more visual tutorial could help alleviate this issue, along with more time to practice with the mouse interface.

- **Cut plane orientation** — In an effort to keep the interface as simple as possible, the option of choosing which side of a cut plane should be removed, was not implemented. However, the user feedback collected made it clear that this option was something most users would like to see implemented.

- **Haptic Feedback** — Haptic feedback has previously been shown to improve accuracy, but not task times, in 3D target acquisitional tasks [118]. None of the test participants seem to notice the magnetic draw to the cut planes, nor was the haptic feedback provided by the pig used extensively.

- **Overall Preference** — Five of the eight participants expressed to prefer the Phantom Omni. Given the nearly identical SUS score, we believe that a significant part of the users preference is based on anticipated performance, given more time with the interface.

- **Phantom Omni Preference** — Almost all of the users who noted their preference for the Phantom Omni, noted it as being easier to navigate three dimensional space with. A single user noted the preference due to all of the necessary functions being operable from one hand.

- **Mouse Preference** — Out of the three users preferring the mouse, two noted that with additional experience they would have preferred the Phantom Omni. A single user noted preference for the mouse based on ease of learning and compact size for mobility.

4.9 Discussion

At first glance, it might seem that the usability test failed to establish the most sought after answer. Which interface did the users prefer? At this point, neither is the clearly correct answer. This result is very significant and should be seen in the light of the brief training period each user was provided with prior to using both interfaces. I believe that the Phantom Omni will, in the long run, be the preferred interface of choice due to its ease of 3D interaction given its six degrees of freedom. But despite it having analogous moving and rotational behavior with the virtual elements, when compared to the mouse interface, it still was not heavily favored, quantitatively, by any means. With the limited sample size, it is hard to say anything significant about what exactly caused both interfaces to quantitatively perform approximately the same. However, given the explicit testimony from several users, that they would prefer the Phantom Omni with further use, at least suggests that the Phantom Omni would become the preferred interface.

Apart from the quantitative evaluation, a lot of valuable information was also gathered from the qualitative evaluation of the test. It is my impression that most pressing usability issue, apart from all the ones mentioned in the previous section, is how to make both the interfaces easier to learn to use. It was quite clear that none of the participants had worked with similar interfaces before, and this was made all the more apparent by their occasional struggles to orient the pig carcass in the way they intended.

The testing also made it clear that the haptic feedback provided by the pig and the cutting planes went largely unnoticed by all of the test participants. None of them commented on it except during the introductory phase where they were shown how haptic feedback worked.

The conclusion is that, although neither interface clearly performed better than the other, both interfaces allowed all of the participants to create all of the products in question. Based on the statements of the participants I would suggest another usability test with both interfaces addressing the usability issues discovered in the current versions. It would be imperative that the participants participate in a longer testing session to familiarize themselves further with both interfaces. As previously noted, I would expect the Phantom Omni to outperform the mouse in the long run, due to its more intuitive interaction style.

Chapter 5

Future Parameterization

In the previous chapter, the interface of the PPP is described in detail along with the design process. The interface performs planar cuts as well as providing future support for customized cutting, using a binary mask. The exploration of more customized cuts and the constraints imposed by the existing standards in the meat industry is explored in this chapter. A perspective of future uses of the PPP is provided in section 5.1.

Hansen explored virtual cuts of all variations in his thesis [32], which are categorized into the following groups:

- **Anatomical Cut** — Cutting along a boundary of the anatomical structure of the pig carcass. Usually along a membrane separating muscles or along a bone.
- **Semi-Anatomical Cut** — Cutting along a boundary of the anatomical structure of the pig carcass, within geometric constraints, e.g. cutting 5 cm along a bone, or trimming a fat layer to a given thickness [88].
- **Geometric Cut** — Performing a cut along a geometric constraint, such as a plane, usually guided by anatomic cues.

The first two types of cuts are typically completed via manual labor in the

abattoir, whereas the last type has become mostly automated. To expand upon the PPPs interface, it would be an ideal next step to incorporate cuts similar to the anatomical cuts completed via manual labor. Given that the PPP supports a haptic feedback device, it might appear like an ideal solution to emulate real world meat interaction. Specifically, allowing the user to guide the knife along muscle membranes, to perform a precise cut, according to established standards. Incorporating haptic feedback would be ideal to keep the user aligned with standardized industry cuts.

However, the goal of the PPP, and any other interface for that matter, is to guide the users to achieve their goal as quickly and efficiently as possible. If established standards can be used to haptically guide the user along pre-set cut paths, there's no real reason to not just let the user pick an established product, and have the system perform all the necessary cuts. In other words, there is no reason to let the user freely cut the products early in the prototyping phase, since most new products rely on already established standards. Customizability requiring manual labor does not enter into the development process until at the very end of the product production cycle.

To properly apply the standardized anatomical cuts to a real world pig carcass, it is important to compensate for the biological variation in the pigs anatomy. Hansen [32] presents research regarding a so-called pig atlas, which can serve as a mean representation of all pigs in a given population, from which individual variations can be derived. As such, it is an ideal starting point when intending to generalized cuts to all pigs.

The next version of the PPP interface, allowing for more custom cuts, should have the interaction structured in the following fashion:

1. **Geometric Cuts** — This phase of product design would be nearly identical to the current implementation of the PPP, except allowing the user to simply select predefined cuts creating the foreend, middle or ham. This division is typically performed on every pig carcass passing through on abattoir.
2. **Product Cut** — The previously chosen third of the pig is cut to conform to a specific standardized product, if needed. The standardized products are defined on the pig atlas in order to let them easily be applied to variations of a pig's anatomy. A properly segmented pig atlas would also allow the user to include or exclude specific bones or muscles, as part of the product.
3. **Custom Cut** — The final product is further customized to meet the customers needs. The cuts required to arrive at this final product are free

for the user to modify and move as needed, in order to make full use of the affordances provided by the virtual product.

The entire process from start to finish should be applied to an established mean pig carcass, allowing for any biological variation of the completed prototypes to be visualized. The estimated lean meat percentage for a variety of complete products could be easily calculated, allowing for an early forecast of potential earnings and losses, during product development.

5.1 Future Integration

The modernization of the slaughterhouses and the abundance of pig carcass data available for use, due to the introduction of CT scanning, opens up a wide array of possibilities. The PPP is just the first step in a long line of potential improvements an interactive application could serve:

- **Interactive cost/profit projection** — Recent research [117] has already paved the way for better estimates of the lean meat percentage (LMP) of pig carcasses using virtual dissection. The PPP is an ideal framework for the integration of a realistic model of the expense each type of cut would incur on the production of the final product. This model would need to evaluate the complexity of the applied cuts to determine which could be completed using the existing automated assembly line, and which would need to be performed by manual labor. The model could also provide an assessment of the byproducts created as a result of the final product and their value.
- **Cut Planning** — The products created by the automated assembly line in a modern slaughterhouse are still quite limited in their variation. In fact, many of the automated systems in the assembly line are built and optimized for the exact purpose of creating identical products despite the biological variation of each pig carcass. The more custom operations are still performed by hand, by trained butchers. An increase in the flexibility of the automated product creation procedure would pave the way for using the PPP as a cut instruction generator. Theoretically allowing the user to create a pre-planned set of cutting instructions, for the machines to execute on any number of pig carcasses.
- **Personnel Training** — The PPP could serve as an interactive training system to educate staff about the anatomy of the livestock used in product

production. It is also possible to use the PPP as a training tool for proper slaughter procedures. However, it is important to note that with the currently available technology, the virtual interaction with a pig carcass, cannot hope to supplant the real world version. Instead, it could aid its users to remember proper cutting procedure and evaluate their applied cuts compared to expert cuts. Effectively making it a useful educational support tool.

Chapter 6

Overview of Contributions

This chapter summarizes the main results obtained in the papers included in part II.

6.1 Customized Texture Transfer function

Chapter 8 presents a novel approach for improving direct volume rendering, using synthesized textures in combination with a custom transfer function. The intention is to narrow the visual gap in between the directly rendered pig carcass volume and a real pig carcass, using a piecewise constant transfer function. The transfer function maps density values to corresponding tissue types, texturizing the volume providing a more realistic appearance. We acquire textures for the three general types of tissue present on a pig carcass; fat, muscle and bone. These pictures are used as input in our implementation of Kopf et als [62] solid texture synthesis algorithm.

The algorithm accepts 2D textures as input (exemplars) and iteratively improves upon a solid texture comprised of random samples from the original input. The texture is improved by optimizing an energy function, which measures the difference between the input exemplar and the texture being synthesized. Differences are calculated by comparing small texture patches (neighborhoods)

from the synthesized texture, with identically sized texture patches from the exemplar. The algorithm uses a variety of methods to accelerate the synthesis including principal component analysis, meanshift clustering, and approximate nearest neighbor search [89].

Despite a few initial failures to generate acceptable solid textures in the case of certain input exemplars, the method yields an acceptable result for each of the tissue types. The textures are applied using a piecewise constant transfer function, mapping voxel intensities to 3D texture volumes.

Direct application of these textures to the solid yields noticeable periodicity. This artifact is reduced via multi level application of the same texture at three distinct levels applied with a measure of transparency. Figure 6.1 visualizes the three scaled applications of the muscle texture, yielding the improved result in the lower right.

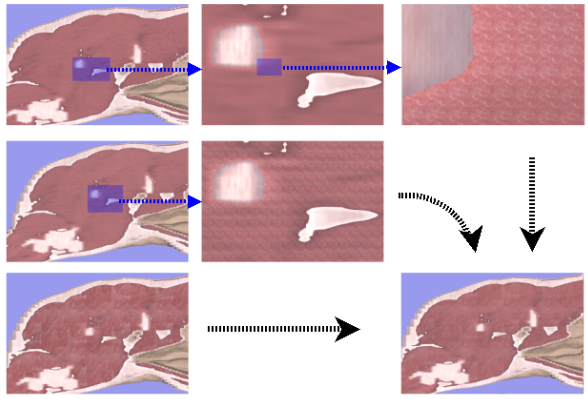

Figure 6.1: Three differently scaled muscle textures combined to create the final result. The top and middle segment show zoomed in areas to show finer detail.

The result is a significant improvement in visualization quality, compared to the simple linear transfer function, as visualized by the comparison in Fig. 6.2.

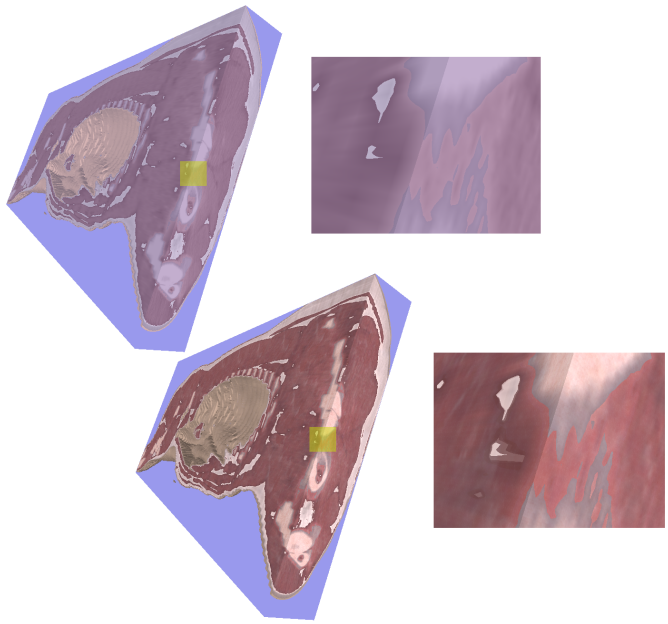

Figure 6.2: On the top, volumetric data from the pig carcass, visualized without enhanced graphics. The colors for the meat, bone, and fat tissue are the average color values of the textures applied on the right. On the bottom, volumetric data from the pig carcass, visualized with enhanced graphics. The highlighted sections in yellow indicate the zoomed section displayed on the right.

6.2 Automatic Quality Measurement and Parameter Selection for Example-based Texture Synthesis

Chapter 9 continues research on the texture synthesis algorithm presented in Chapter 8, by examining direct and indirect methods to automatically select parameters for example-based texture synthesis.

Hong et al. [54] present a method with which to directly estimate the scale in a texture, by representing textured patches as probability density functions. We apply the method to textures utilized by Kopf et al. [62] and receive mixed results, as visualized in figure 6.3. The scale measure presented by Hong et al. discards spatial location information which leads to the inability to properly detect all structural elements. The detected scale is arguably incorrect when

comparing the last two textures in figure 6.3. The bigger brick texture (e) is detected as having the smaller scale compared to the smaller brick texture (f). We present novel alternative methods of measuring scale, based on clustering and tree-based representations.

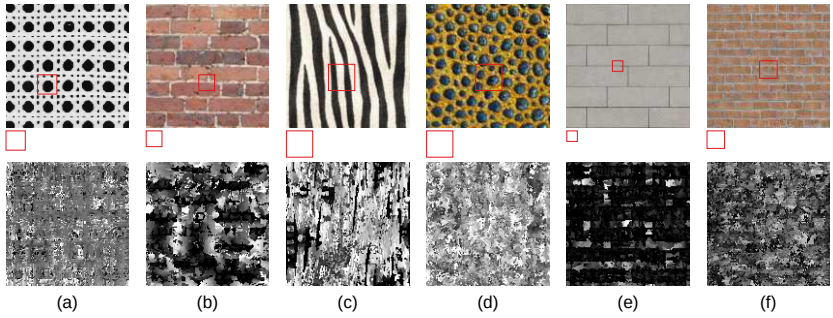

Figure 6.3: The top row shows the original textures, along with the median neighborhood estimated from every single pixel as a red box in the texture, using the approach by Hong et al. [54]. The same neighborhood size is also visualized immediately below the texture. The bottom row is a scale map representation of each of the textures, obtained by applying the energy equation by Hong et al., where each pixel is given an intensity matching the estimated best neighborhood size surrounding that pixel. The more intense the pixel, the larger the estimated neighborhood for that location.

We also present and test a set of heuristics which, combined with an objective similarity measure, are capable of successfully synthesizing a better result than those using a standard set of parameters (defined by Kopf et al.), as pictured in Fig. 6.4.

Automatic parameter selection is a vast problem, and the research presented in Chapter 9 only covers a small portion of the problems that need to be addressed. The methods presented are not without fault, and the similarity measure occasionally produces failure cases. However, the resulting synthesis using automatic selected settings outperforms the standardized settings provided by Kopf et al. in the majority of cases we tested. We cover all of the major limitations of our research to provide a solid basis for future research as well as a number of methods useful in the next step required to solving the remaining problems.

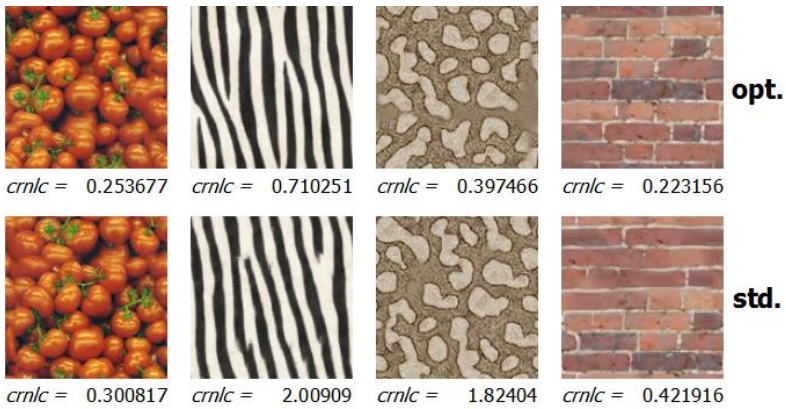

Figure 6.4: Comparison of the four sample textures synthesized using automatically detected optimal settings (top row) and standardized settings used by Kopf et al. (bottom row). The quality as measured by the crude *reverse neighborhood lookup comparison* test is listed below each result. The optimized approach results in a better synthesis in 3 out of the 4 presented textures. The brown texture (third from the left) is a failure case, showing a slightly more blurry texture than the standard approach.

6.3 Real-Time Registration Based Volume Interpolation

Chapter 10 continues expanding on upon research, presented in Chapter 8, aimed at improving the visual quality of directly rendered volume data. Direct volume rendering commonly relies on the raw computational power provided by modern GPUs in order to achieve real-time visuals. The volume itself is often anisotropic as a result of the process used to acquire it, and the rendering process traditionally relies on interpolation, in order to turn the discrete (volume) signal into a continuous one.

Modern graphics hardware only provides built-in functionality supporting linear interpolation. Other types of interpolation can be implemented via software-based shaders, however the performance penalty is often severe due to the high number of texture look ups involved.

We present a novel approach for real-time anisotropic volume data interpolation on a GPU and draw comparisons to standardized interpolation alternatives. Our approach uses a pre-computed set of cross-slice correspondences to compensate for missing data. The method for calculating correspondences is based on work by Ólafsdottir et al. [69] who expand upon previous work regarding registration-

based interpolation by performing two-way registration.

We perform a quantitative and qualitative analysis of our approach, by evaluating the results of its application to two different volumetric data sets. The primary data set used for testing was derived from an isotropically scanned pig carcass consisting of $212 \times 512 \times 1662$ voxels. Using this data set as the ground truth, we created increasingly sparse versions of the data set (removing slices on the z-axis) and applied the registration-based interpolation. We investigated both visual quality, as well as numerical divergence from the ground truth. Applying the method on increasingly sparse data sets, we gather valuable data revealing performance boundaries of the approach.

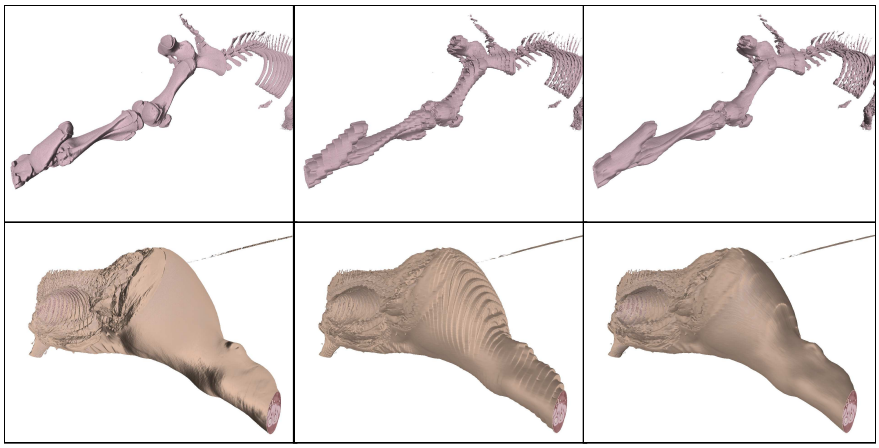

Figure 6.5: 3×2 comparative screenshots of the (originally) isotropic data set of a pig carcass. On the left, the unaltered original isotropic data set. In the middle, a sparse version of the same data set retaining every 9th slice, interpolated using linear interpolation. On the right, the same sparse data set interpolated using real-time GPU accelerated registration-based interpolation. While the interpolated version lacks the visual fidelity of the isotropic data set, due to smoothed surface normals, it looks much better than the linearly interpolated anisotropic data set in the middle.

Our method produces high quality interpolation, as seen in Fig. 6.5, with a moderate performance impact compared to alternatives. The rendering performance impact compared to linear interpolation is visualized in Fig. 6.6. Although the performance of the registration based interpolation is expectantly lower than the built-in hardware interpolation, it still yields interative frame rates. At a sparsity level of 6, it performs approximately as well as the trilinearly interpolated isotropic pig, while using approximately half of the memory capacity. The visuals are comparable, although the isotropic pig has a higher fidelity given its

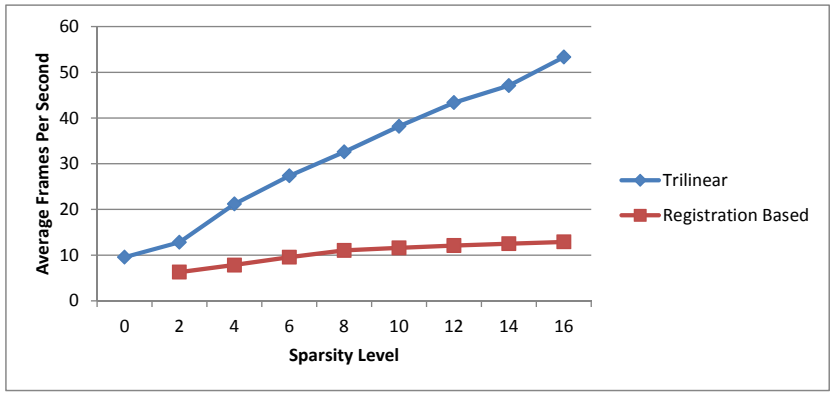

Figure 6.6: Average frames per second, measured over a 60 second time period, for both hardware supported trilinear interpolation, and registration based interpolation. Both types of interpolation show a linear increase as the data set becomes more sparse. Interesting is the fact that at sparsity level 6 the registration based interpolation equals the trilinearly interpolated isotropic data set.

non-interpolated surface normals.

To summarize, the presented registration based method is ideal for reconstructing sparse data sets, and allows for high quality rendering on memory limited hardware such as mobile graphics cards.

6.4 Pig Product Prototyper: Cutting interface design

Chapter 11 presents the culmination of most of the aspects of this thesis. The conception and design of the pig product prototyper is described, along with the usability study performed to evaluate its performance.

The PPP is a unique interface, explicitly designed for the purposes of prototyping pig products and provides both a mouse and keyboard based interface, as well as a Phantom Omni based interface. Collaborating with DMRI [28] and Danish Crown [26], the interfaces were compared in a comparative usability study with eight volunteers from the target user group.

Prior to the usability test, an expert butcher is consulted to produce five industry standardized pig products. The accuracy of these products confirm that the

directly rendered volume data is adequate, in allowing an expert to recognize the pigs internal anatomy, and realistically recreate existing products.

A total of eight volunteers from the targetted user group participated in the usability study. Each user was given a brief introduction to both interfaces, and posed a series of basic navigational challenges to improve their skill with the respective interfaces. The user is then tasked with re-creating five industry standardized products. Their completion times are tracked and final results both objectively and subjectively compared to the previously established product ground truths.

The usability test reveals no major usability issues with either interface, and the quantitative evaluation did not favor one interface over the other. None of the volunteers have had previous interactions with either of the interfaces, nor the Phantom Omni controller. The subjective opinion of the volunteers leaned towards preferring the Phantom Omni. The most prominently cited reason was the direct correlation of moving and rotating the Phantom Omni, and the respective movement and rotation of virtual elements.

Due to this direct correlation in interaction using the Phantom Omni we believe that it is just a matter of further experience with the interface, leading to it being significantly preferred.

Chapter 7

Conclusion

This chapter summarizes the contributions of this thesis, published in Part II with theoretical background and expanded elaboration in Part I. Perspectives are drawn in relation to the original objectives, stated in section 1.1, along with some closing remarks.

7.1 Summary

The first objective of the thesis is to study and apply methods for realistically rendering pig carcass volume data. Volume visualization is a key ingredient in moving pig product prototyping into the digital realm, where products can be designed with affordances not provided by real pig meat. Apart from the importance of recognizing pig anatomy, it is beneficial to make the visualization as realistic and aesthetically pleasing as possible. The results of efforts to achieve this goal are presented in Chapters 8, 9 and 10.

We address improving direct volume rendering and related visualization in three ways. First, we improve the appearance information of the volume data by using a custom transfer function mapping density values from CT scans to synthesized textures, created from high quality images of muscle, fat and bone tissue, from a

pig carcass. Periodicity produced by repeating volumetric textures is mitigated by multiple transparent layers.

Second, we analyze the problem of texture synthesis parameter selection, for the purposes of creating higher quality results while automating the process. We base our work on a state-of-the-art texture algorithm, namely Texture Optimization presented by Kopf et al. [62]. The standard parameters provided by Kopf et al. generally produce good results. However, many textures perform better with optimized settings, which we estimate using an indirect approach. We use a similarity measurement, to search for the optimal texture synthesis parameters, by maximizing the quality of the synthesis, as a function of parameters. Together with a set of heuristics we create a foundation for improving automatic parameter selection. We also analyze and present indirect methods of estimating a limited set of parameters. The direct method does not rely on a similarity measure, and is less biased. However, further research is required to determine their viability.

Third, we propose a novel real-time interpolation method, for improving anisotropic volume data rendering. The method is based on registration-based interpolation by Ólafsdóttir et al. [69] and provides significantly improved results compared to standard interpolation techniques. The method is capable of compensating for considerable sparsity in the volume data, as it utilizes structural information to improve interpolation.

The second objective of this thesis is to develop an interactive haptic-enabled system for the purposes of simplifying and improving the existing communication process, when developing pig product prototypes.

Through a collaborative effort together with DMRI [28] and Danish Crown [26] we developed the Pig Product Prototyper. The application allows the user to commit planar cuts to a volumetric visualization of a pig carcass using either the standard mouse or Phantom Omni haptic feedback device.

The design process started by communicating with experts and users experienced in product prototyping. Having established key needs, we designed the interface using a set of heuristics and a trial and error approach. Once the interface reached the significant final milestone a thorough usability test was conducted with the target user group.

The formative comparison of the two input devices did not yield quantitative data heavily favoring either of the interfaces. However, the data revealed comparable performance, despite users past experience with the mouse controller. Qualitative analysis of the users perceived usability of both interfaces were aver-

age, indicating that both still have room for improvement. Subjective feedback gathered from users leaned towards using the Phantom Omni, with the most often cited reason being its direct 1-to-1 correlation of input and on-screen reaction. Seven out of eight users stated that they favoured the Phantom Omni outright, or would do so with more practice.

7.2 Conclusion

Apart from the individual contributions this thesis makes to the research fields of volumetric visualization and human-computer interaction, the primary achievement is the culmination of work into an interactive meat product prototyping tool. Valuable lessons are learned only from the interplay in between the developed technologies, revealing which contributions are important, and which less so.

Enhancements to volumetric visualization were originally considered very important for the succesful integration of the tool. Not only as a means of impressing prospective clients with realistic rendering, but also necessary for users to better recognize pig anatomy. Although the visual improvements have contributed to more realistic and aesthetically pleasing visuals, the significance of their improvments lie, not in realism, but functionality.

Volumetric texture mapping using solid textures is a viable method for more realistic rendering. The real importance lies in the minimum resources required to apply the technique itself. Automated texture synthesis parameter selection contributes to reducing this workload even more. The functionality these methods provide is a reduction in complexity for people while still achieving more realistic rendering.

Registration-based volume interpolation also improves the visual appeal of the rendered pig carcass. It is capable of rendering anisotropic data sets at interactive framerates and reconstructs smooth surfaces from very sparse data sets. But its significance is perhaps biggest when seen from the perspective of the meat industry. Improved data reconstruction makes it possible to produce realistically rendered visuals using fewer data slices. Fewer data slices leads to quicker CT scans, which is crucial for future integration.

Ironically, this means that the most significant contributions of the improved visual quality, is not improved visual quality, but the affordances they provide.

The interactive product prototyping tool was initially conceptualized as a type

of simulator. Utilizing modern technology, this software could potentially serve to replace real-life interaction with a better, more usable, virtual counter-part. The real revelations during development was which aspects of this simulation were actually beneficial, and which were not.

For instance, haptic interaction is a fascinating technology. Users are eager to try new experiences and even crude haptic interaction is appreciated. The Phantom Omni was originally considered due to its haptic capabilities, but its most useful functionality turned out to be its six degrees of freedom. It allowed for the analogous application of movement and rotation made by the user, directly to a given virtual object. The object would move and rotate in the same manner the Phantom Omni pen was moved. Arguably, as transparent an interface as possible without having the user physically move the virtual object.

Advanced interactive concepts using haptics ended up restricting the user more, instead of simply limiting interaction to relevant actions. It seems obvious in retrospect, but the simplest solution turned out to be the best.

The usability test confirmed that the tool enables sales personnel from the target group to reconstruct standardized meat products. An expert user could also easily identify anatomic cues, and use these to produce highly accurate reconstructions with which to compare our results. This is a positive outcome and bodes well for the future integration of the tool, in the meat industry.

I believe the developed Pig Product Prototyper is a useful tool, and its integration into the current meat industry work flow improves communication due to the affordances it provides. The possibilities are only increased as future functionality is added to potentially support personnel training, robot cut planning, and interactive product profit projection.

Part II

Contributions

Chapter 8

Anisotropic 3D texture synthesis with application to volume rendering

Lasse Farnung Laursen, Bjarne Kjær Ersbøll
Jakob Andreas Bærentzen

Abstract

We present a novel approach to improving volume rendering by using synthesized textures in combination with a custom transfer function.

First, we use existing knowledge to synthesize anisotropic solid textures to fit our volumetric data. As input to the synthesis method, we acquire high quality images using a 12.1 megapixel camera.

Next, we extend the volume rendering pipeline by creating a transfer function which yields not only color and opacity from the input intensity, but also texture coordinates for our synthesized 3D texture. Thus, we add texture to the volume rendered images. This method is applied to a high quality visualization of a pig carcass, where samples of meat, bone, and fat have been used to produce the anisotropic 3D textures.

8.1 Introduction

The use of volumetric data is becoming increasingly common within research fields such as medical visualization, food production and graphics. This data is also ever increasing in size as the scanners providing the data, e.g. CT, MRI, and

ultrasound scanners, are improving and thus able to provide higher resolutions. Increased precision and more detail is a natural evolution as having too much information, is somewhat of a luxury problem.

When concerned with rendering volumetric data in real time, two issues persist. Firstly, the features that we would like to visualize might be on a finer scale than the voxels, despite the ever increasing amount of volume data. In our case, we visualize pig meat, and the variation in the texture of pig meat is on a finer scale than the resolution of our CT scan. Moreover, the voxels in our CT scanned data are stretched ten times along one axis. This problem is compounded by a second issue which is the fact that the CT intensities represent material density, which is not directly related to the appearance of the underlying tissue.

We present a novel approach which aims to alleviate both issues. With prior knowledge about the type of volumetric data we wish to visualize, we synthesize an anisotropic 3D texture which is applied to the volume data via a customized transfer function. Using this transfer function, we map the CT intensities to a high resolution solid pig meat texture which gives a qualitatively far better representation of the meat than any single color. Moreover, the solid texture texels are not stretched.

Solid textures are an ideal fit when rendering volumetric data. In almost all cases, there is an interest in rendering what is beneath the surface or subdividing the data to expose some deeper layer. Since a solid texture shares the same number of dimensions as common volume data, its application is relatively straightforward.

8.2 Related Work

The focus of this paper can be divided into solid texture synthesis, and the application thereof in volumetric rendering.

8.2.1 Solid Texture Synthesis

Considerable work has been done within the field of texture synthesis, from parametric methods [52] to non-parametric methods [27, 51], as well as alternative approaches [127]. Most texture synthesis algorithms use a sample texture as input, referred to from here on as exemplar. This exemplar forms the basis for either a parametric model, which synthesizes a new texture based on modeled

parameters, or for a non-parametric algorithm, which reuses elements from the exemplar and recombines these to create a new, yet similar, texture.

Solid texture synthesis has been pioneered and expanded upon within the past two decades. Several methods, both parametric [39] and non-parametric [123], as well as alternate approaches [56], have been presented.

A recent texture synthesis method, which we use to create our anisotropic textures, is called texture optimization [63, 66]. This method iteratively improves the texture as a whole, making each modification smaller and more refined.

8.2.2 Volumetric Transfer Function

Volume rendering [31] has come a long way. Most applications today make use of graphics hardware to improve performance [25]. The field has seen a dramatic increase of research into all kinds of visualization techniques involving volumetric data. Most volumetric data originates from either computed tomography or magnetic resonance scans, which do not yield a direct mapping to appearance attributes (i.e. color and texture of the scanned tissue). An obvious field of research is therefore to provide proper color and texture to this otherwise appearance deficient data. The visible human project is one such example, where a male and female body has been scanned, and subsequently cut and photographed to obtain the correlation between density and appearance. One method with which to color the data, is the use of a transfer function [44].

Many methods for creating transfer functions exist. From a simple pre-defined function capable of transforming between two number domains, to a user defined transfer function allowing for iterative refinement through user input [23]. In most cases, user input is desirable since the transfer function is often used as a tool to highlight or hide specific features in the volume data.

Other approaches include Dong and Clapworthy [29], who use 2D input exemplars to apply and synthesize texture to a volumetric volume simultaneously. By analyzing the orientation of each voxel in the volume data a patch based synthesis strategy is applied to apply and expand the 2D exemplar to the volume.

Lu et al. [79] expand upon an existing 2D synthesis algorithm to create a flexible system for volume illustration. By extending the concept of Wang Cubes into the third dimension Lu et al. create a tileable solid texture set.

Manke and Wünsche [82] provide a formal framework for applying solid textures

to a volume, similar to the work in this paper. They also present methods for dealing with discontinuous mapping. In contrast to this paper, however they do not touch upon the scaling or periodicity issues of applying a repeating solid textures to a volume.

In this paper, we use a simple, piecewise constant transfer function which maps voxel intensities to entire texture volumes, similar to Manke and Wünsche [82]. Subsequently, the color values at the given position in the volume are obtained by lookup in these texture volumes. The voxel density is used as an indicator for opacity. The textures are applied in a multi-scale fashion to minimize the periodicity, which is further described in Section 8.6.

8.3 Overview

It has been our overall goal is to improve the visualization of CT scanned data. By applying a solid texture to the data via a transfer function, we are able to increase the visual detail at a minor cost to the computations required.

We employ the texture optimization method presented by Kopf et al. [63], to synthesize our anisotropic textures. There is a large overlap with our description and [63]. This is partly to highlight particular details of our implementation and partly to make the present paper more self-contained.

Unfortunately, the aforementioned texture synthesis method does a poor job of synthesizing textures with only low frequency features. This leads to some muscle textures being comparable to base noise textures with similar colors.

Due to computational limitations, synthesizing solids larger than 128x128x128 is not feasible. This presents a number of scale and periodicity issues which we explore in sections 8.6 and 8.7. In short, we apply the synthesized texture in multiple scales to allow for fine and rough effects. We still make use of the CT data to add additional rough detail.

The results of these iterative improvements are compared and discussed, also in section 8.7.

8.4 Solid Texture Synthesis

As previously explained, texture optimization is an iterative method where the difference between the input exemplar and the synthesized solid is minimized. The difference is measured by a global texture energy function which compares fixed sized 8x8 2D neighborhoods. For now, let us assume that each voxel/texel defines its own neighborhood. We define a simplified global texture energy function, similar to the one by Kopf et al. [63]:

$$E(N_s, N_e) = \sum_{i=1}^{c} \|N_{s,i} - N_{e,best}\|^r. \tag{8.1}$$

The neighborhoods in the synthesized solid and input exemplar(s), are denoted by N_s and N_e respectively. The total number of number of neighborhoods from the synthesized solid (N_s) is denoted c. The i'th vectorized neighborhood in the solid is denoted $N_{s,i}$, and its closest match (in L_2 norm) in the input exemplar(s), is denoted by $N_{e,best}$. The exponent $r = 0.8$ makes the function more robust against outliers [63, 66].

Initially, the synthesized volume is comprised of randomly selected texels from the input exemplar. The volume is then iteratively improved to resemble the input exemplar(s). The process is comparable to an expectation maximization algorithm. We first find the "best looking" parameters, then we optimize based on those findings, and repeat the process.

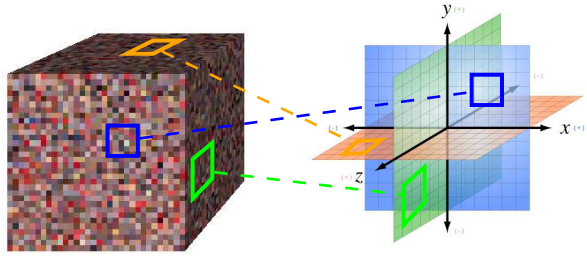

Figure 8.1: Exemplars on the three planes orthogonal to the main axes.

As mentioned previously, comparing the synthesized texture to the input exemplar(s) is done by comparing fixed sized `8x8` neighborhoods. These neighborhoods are extracted from both the synthesized volume and the input exemplar(s). However, there is not - as previously mentioned - one neighborhood

assigned to each voxel. Rather, each voxel is indirectly related to the neighborhoods that includes it.

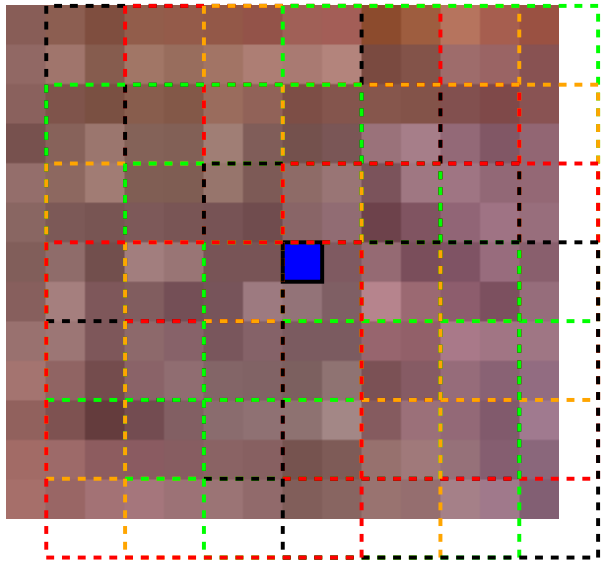

Figure 8.2: Density of neighborhoods on both exemplar and synthesis textures.

On the input exemplars, these neighborhoods lie on a densely populated grid, since we want to use all the available information provided to us, about the texture to be synthesized. In the synthesized volume the neighborhoods lie on a sparse grid (spaced 1 voxel apart like Kopf et al. [63]), and only on planes orthogonal to the three main axes of our coordinate system, as shown in Figure 8.1. This serves to reduce computation time and avoid re-sampling issues.

Figure 8.2 visualizes the sparse grid upon which the synthesized neighborhoods lie. A given voxel - highlighted in blue - is a member of 16 on any given plane, due to the synthesized solids toroidal boundary conditions.

Once the all the neighborhoods have been extracted, the "best looking" parameters are then found by locating the least different neighborhood in the input exemplar(s), for each neighborhood in the synthesized volume. Once found, each voxel is assigned a new value based on texels in the corresponding best matches of the neighborhoods overlapping that voxel. Essentially averaging all the contributions to make a new color:

$$S_v = \frac{\sum_{s_v \in N_s} t_{N_{e,best}}}{c_{s_v}}. \tag{8.2}$$

The new color assigned to the voxel in the synthesized solid, denoted s_v, is an average of several existing color values. The above equation states that for each neighborhood N_s the voxel is a member of, we find the matching texel t, in the best matching neighborhood $N_{e,best}$. This sum is finally divided by c_{s_v}, which denotes the number of neighborhoods the voxel s_v is a part of.

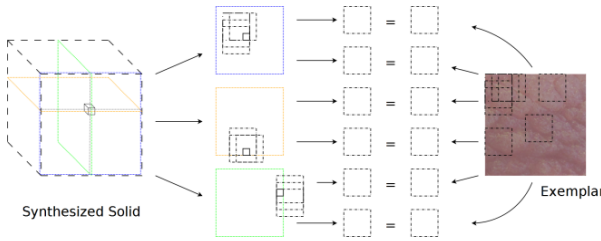

Figure 8.3: Neighborhoods on the three planes orthogonal to the main axes matched to input exemplar neighborhoods.

Figure 8.3 visualizes equation 8.2 in practice. In our sparsely populated grid on the synthesized solid, a single voxel is member of 16 neighborhoods on a single plane orthogonal to a main axis. Since we have three such planes, visualized as blue, yellow, and green in figure 8.3, the voxel is a member of a total of 48 neighborhoods. Each of these neighborhoods has a corresponding match in an exemplar. The texel overlapping the same position in each of these neighborhoods contributes to the sum, which is eventually divided by the total number of contributions (in this case 48), yielding the new color.

The optimization algorithm is actually performed on multiple levels of differing quality. The synthesized solid initially consists of `32x32x32` voxels, and input exemplar(s) are scaled to `32x32` respectively. Once the synthesis process reaches certain conditions, outlined in section 8.4.5, the volume is scaled up to `64x64x64` using trilinear interpolation. Due to computational restrictions of performing a nearest neighbor search in a high dimensional space, the synthesis is only performed up to a resolution of `128x128x128`.

8.4.1 Approximate Nearest Neighbor

In a standard-RGB texture, an `8x8` neighborhood consists of 192 values. Finding the nearest neighbor in a 192 dimensional space is a computationally expensive operation.

We apply the same optimizations as Kopf et al. [63] to reduce the computation complexity. A principal component analysis is performed on the neighborhood vectors from the exemplar(s). By only preserving the coefficients required to maintain 95% of the variance, we can typically reduce the number of dimensions by half, or more.

We also employ the ANN: Approximate nearest neighbor library [89]. The library accepts a value $\mathcal{E}$, and returns an approximate nearest neighbor guaranteed to be at most $\mathcal{E}+1$ away from the true nearest neighbor. We employ $\mathcal{E}=2$ as dictated by Kopf et al. [63].

8.4.2 Weighting Scheme

As previously mentioned in section 8.4, using an exponent of 0.8 in equation 1, causes it to be more robust against outliers. However, minimizing the L_1 norm is more cumbersome than minimizing the L_2 norm. So instead we introduce a weight into the equation and rewrite the terms of the energy function (1) to the following (similar to Kopf et al. [63]):

$$\begin{aligned} \|N_{s,i} - N_{e,best}\|^r &= \|N_{s,i} - N_{e,best}\|^{r-2}\|N_{s,i} - N_{e,best}\|^2 \\ &= \omega_{e,best}\|N_{s,i} - N_{e,best}\|^r. \end{aligned} \tag{8.3}$$

where $\omega_{e,best} = \|N_{s,i} - N_{e,best}\|^{r-2}$. This leads to the following quadratic formula which we seek to minimize:

$$E(N_s, N_e) = \sum_{i=1}^{count(N_s)} \omega_{e,best}\|N_{s,i} - N_{e,best}\|^2. \tag{8.4}$$

Equation 4: Improved energy function.

The weight parameter $\omega_{e,best}$ makes sure that the exemplar neighborhood closest to a given synthesized neighborhood, carries the most weight. Instead of a

straight average as applied in equation 8.2, we are now calculating a weighted average which leads to the following formula when calculating a new voxel value:

$$S_v = \frac{\sum_{s_v \in N_s} \omega_{e,best} t_{N_{e,best}}}{\sum_{s_v \in N_s} \omega_{e,best}}. \tag{8.5}$$

Instead of dividing the sum by the total number of contributors, we now divide by the total amount of weight distributed among the contributions.

8.4.3 Meanshift

Although adjusting each contributing texel with a weight parameter yields better results and speeds up convergence, there are still numerous textures which fail to produces acceptable results. One persisting issue is that outliers still contribute to the final result, even if their contribution is minimal.

In order to minimize contribution from outliers, Kopf et al. [63] employ a clustering approach, proposed by Wexler et al. [128]. In short, every contributing texel is considered to be a cluster. These clusters are then merged depending whether their center is within a distance of τ to one another. If any new clusters emerge, the process of searching and merging is repeated, until no further clusters form. Only texels from the dominant cluster end up contributing to the new voxel value.

The threshold τ is decreased with each iteration over the course of a single resolution level convergence. Once the synthesized texture converges on a single level, the thresholding value τ is reset. We found that setting $\tau = 10$, $\tau = 0.05$, and $\tau = 0.01$ worked well in many cases, on the lowest, medium, and highest resolution level, respectively.

8.4.4 Histogram Matching

The previously mentioned modifications to the original synthesis method, speeds up convergence and minimizes the impact of outliers. However, the algorithm will occasionally converge at certain minima, which fail to make full use of the exemplar(s) details.

Kopf et al. [63] address this issue by utilizing histogram matching. The weight each texel carries is further adjusted, based upon whether its contribution will

increase, or decrease, the similarity between the histograms of the input exemplar, and the synthesized solid. Practically, they achieve this by keeping track of a 16-bin histogram for each of the input exemplars' channels. Usually, this is just the red, green, and blue channel. Kopf et al. also note the importance of keeping this histogram up to date during each "maximization" phase. Otherwise, the method will just overshoot the intended histogram and overcompensate in the following iteration.

When synthesizing anisotropic textures we maintain one histogram per input exemplar. We let each contributing texel pull in the direction of its exemplars histogram, which seems to work well. Just like Kopf et al. we also traverse the voxels in a random order, to avoid any directional bias.

Histogram matching is an integral part of creating the best results possible via texture optimization. It makes the algorithm take global statistics into consideration while still allowing for the use of a small neighborhood window. Histogram matching also speeds up convergence significantly.

8.4.5 Synthesis Convergence Conditions

We found that a fixed number of iterations yielded the best result with most textures (J. Kopf, pers. comm.). Iterating 100 times on the lowest resolution, 20 on the next level, and 10 on the highest level, worked well with most textures.

8.5 Exemplar Acquisition

As with every other texture synthesis method, we require exemplars of the texture we intend to synthesize. Our exemplars were obtained using a 12.1 megapixel camera, Canon IXUS 120IS, in a well lit setting. Originally, we intended to obtain samples using a multispectral color and texture measurement vision system. This system measures up to 20 different bands across the visible and non-visible spectrum. These precise measurements are then combined to a final standard-RGB image. However, most household cameras actually yield more vivid and realistic colors as each sensor integrates a wider range of the spectrum than the more precise instrument.

8.6 Rendering

To visualize the volumetric data, a simple ray casting technique [45] is applied using the GPU. To obtain the start and end point for each ray, two rendering passes are performed of a cube showing the front- and backface respectively. The cube acts as our rendering proxy and yields the start and end position for each ray, which is recorded into a buffer using the fragment shader.

An additional rendering pass is then performed where the fragment shader traces a ray through the space enclosed by the cube. The ray is traced with 0.001 increments in relation to the unit cube around the volume, and accumulates more color and opacity as it traverses the volume. The "ray-color" starts off black and completely transparent. For each step through the volume, the current density is classified as air, skin, fat, meat, or bone, according to the Hounsfield scale [41]. Its contribution to the overall "ray-color" as well as "remaining transparency" is calculated by the following formulas:

$$\begin{aligned} R_{rgb} &= R_\alpha * l * D_{rgb} * D_\alpha \\ R_\alpha &= R_\alpha * (1 - D_\alpha) \end{aligned} \tag{8.6}$$

The contribution added to the existing color and transparency value of the ray is denoted as R_{rgb}. The amount of contributing light via simple lambertian shading [46], is denoted l. The contributing color and transparency from the classified density is denoted D_rgb and D_α respectively. When "ray-color" is completely opaque, the ray traversal is stopped.

Setting D_{rgb} in Equation 8.6 to the color from the appropriately scaled solid texture produces a result with significant periodicity at high magnification and almost uniform color at low magnification (because of mipmapping). This can be seen in Figure 8.5, in the top and bottom left. To ameliorate these issues, we combine the texture at three levels of scaling, and D_{rgb} is computed as illustrated in the next equation.

$$\begin{aligned} D_{rgb} &= \frac{D^1_{rgb} f^1 + D^2_{rgb} f^2 + D^3_{rgb} f^3}{f^1 + f^2 + f^3} \\ i_{rgb} &= D_{rgb} * int - t_{threshold} \\ D_{final} &= D_{rgb} + i_{rgb}. \end{aligned} \tag{8.7}$$

The color contribution consists of three differently scaled textures D^{1-3}_{rgb} and an associated weight factor f^{1-3}. For each tissue, the scales differ approximately

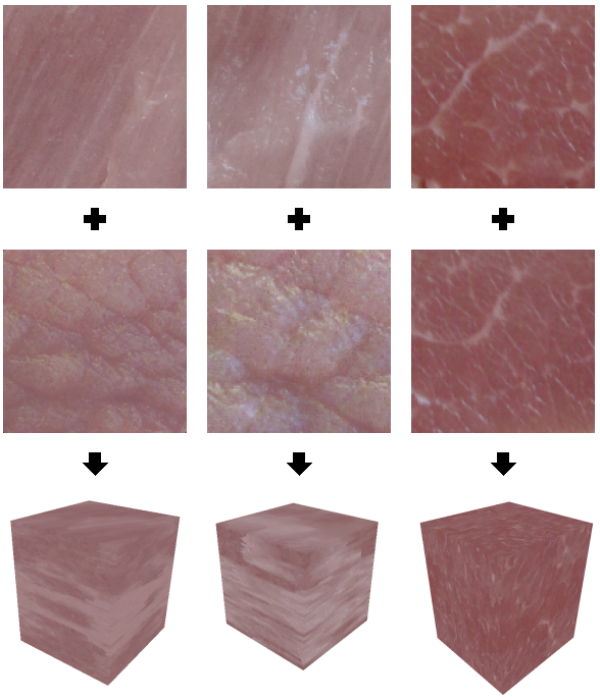

Figure 8.4: Three synthesized solids and their two input exemplars (pig muscle tissue). The left and middle synthesis' yield an unsatisfactory result.

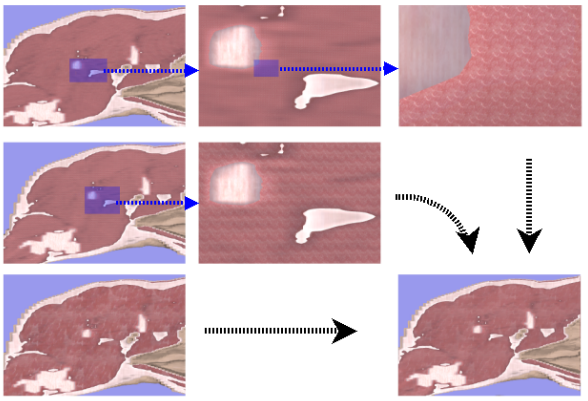

Figure 8.5: Three differently scaled muscle textures combined to create the final result. The top and middle segment show zoomed in areas to show finer detail.

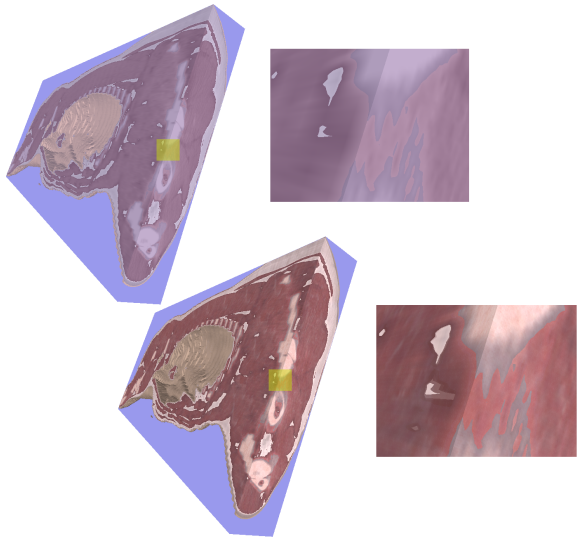

Figure 8.6: On the top, volumetric data from the pig carcass, visualized without enhanced graphics. The colors for the meat, bone, and fat tissue are the average color values of the textures applied on the right. On the bottom, volumetric data from the pig carcass, visualized with enhanced graphics. The highlighted sections in yellow indicate the zoomed section displayed on the right.

a factor of 10, and the weight factor is always highest for the macro texture (approximately 3 to 1). Density is contributed to the final color value D_{final} via i_{rgb}. The value int represents a scaled measure of the density at that point in the volumetric data, and $t_{threshold}$ denotes the density threshold of the contributing tissue. The result of combining the three synthesized muscle textures, along with the density modifier, is visualized in Figure 8.5.

An exception to the calculation outlined in Equation 8.7 is the skin color contribution which yields a constant average color of sampled pig skin, permeated by simplex noise [97] to give some variation to the surface.

The transparency value for either fat, muscle or bone is calculated via the following formula:

$$D_\alpha = max(0.3, int + 0.25). \tag{8.8}$$

Skin has a constant translucency of `0.75`, and air is completely transparent.

8.7 Results

We implemented the method described in this paper entirely in C++. The time required to generate a 128^3 solid depends primarily on the size and richness of the input exemplars. On an Alienware m17x model (using only a single core) the synthesis of our three tissue types would usually converge after approximately 2-3 hours. It was our experience that the algorithm generated the best textures when only forcing two of the three dimensions to conform to input exemplars, regardless of whether isotropic, or anisotropic synthesis.

As previously mentioned in section 8.3, the synthesis algorithm has trouble synthesizing 3D textures based on input exemplars with primarily low frequency features. The two initial attempts in figure 8.4 show how the final synthesized solid ends up looking almost nothing like the original two textures used as input. The third synthesized solid is much more promising.

A question of scale arises when choosing how much surface a single exemplar should cover. To ensure we acquired as homogeneous a sample as possible, and preserve detail, we chose to use exemplars covering a small area of approximately `2x2` cm.

As previously mentioned in section 8.6, the final color value of a voxel is modified

by a simple mapping of the current density. The density modifier allows for a number of low frequency details to show, as is visualized in Figure 8.7.

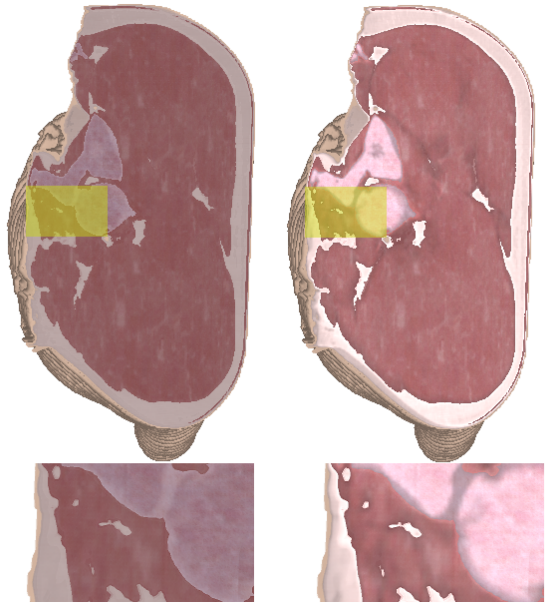

Figure 8.7: Two hams, with and without density value modification. The highlighted sections in yellow indicate the zoomed section displayed on the bottom.

The final result of the visualized volume data with all the aforementioned techniques applied is shown in Figure 8.6.

We perform a preliminary benchmark of the applied synthetic textures by rotating the volume one complete turn, around the y-axis, as seen in Figure 8.8.

	Std. Graphics	Enh. Graphics
Min. Fps	55,3125	56,875
Max. Fps	46,84566889	46,89826405
Avrg. Fps	63,77933111	66,85173595

Table 8.1: Preliminary performance measurements.

The application of the synthesized textures only requires three additional texture lookups per visualized voxel. Since texture lookups are implemented on the hardware level, it comes as no surprise that the performance loss is minimal, as seen in Table 8.1. However, creating the synthesized textures is another matter.

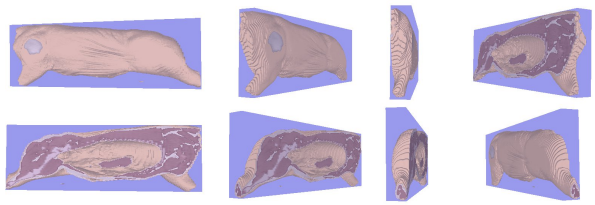

Figure 8.8: The volume data rotation pattern of the preliminary benchmark. Unenhanced pig visualized.

Due to the complexity of a nearest neighbor search in a high dimensional space, performing the synthesis in real-time is an impossibility.

8.8 Conclusions and Future Work

We have utilized an existing texture synthesis approach to produce three anisotropic textures, which were then applied to volumetric data via a custom transfer function, improving upon the original colorless data.

The technique can potentially be applied to any type of volumetric data and is not necessarily constricted to organic tissue.

Although the result has improved significantly, there is still room for improvement.

Our light model is simplistic. Better modeling of how light and meat interact would be an obvious next step since, recently, techniques for real-time interactive computation of translucent surfaces have started to appear, e.g. [122].

As mentioned in the previous section, we modify the final color slightly via the density of the volumetric data. While this adds significant detail to the final visualization, it also introduces a number of artifacts introduced by the scanning method. Figure 8.9 shows how the data acquisition rays from computed tomography leaves visible artifacts in the volume data.

Due to time required to perform a complete solid texture synthesis it could be advantageous to create a larger pre-computed library of multiple tissue types (in addition to the three described in this paper).

Figure 8.9: A close up of the visualized volumetric data showing computed tomography artifacts.

Theoretically, it would also be possible to synthesize in-between textures by using an input exemplar from each tissue type, to smooth the transition between them. A few practical experiments are required to see how convincing the resulting solid textures would be.

It would also be interesting to implement and compare the technique demonstrated by Lu et al. [79]. Using their extension to the wang cube model is also a way of avoiding periodicity in the applied texture.

8.9 Acknowledgments

We would like to thank Johannes Kopf for the invaluable correspondence during the development of this paper. We also extend our thanks to the anonymous reviewers in helping us improve on the paper. This research was supported in part by the Danish Meat Research Institute. The CT scan of the pig carcass was also kindly provided by the Danish Meat Research Institute.

Chapter 9

Automatic Quality Measurement and Parameter Selection for Example-based Texture Synthesis

Lasse Farnung Laursen, Line Harder Clemmensen, Jakob Andreas Bærentzen
Takeo Igarashi, Bjarne Kjær Ersbøll

Abstract

Texture synthesis algorithms have been researched extensively in the past decade. However, most synthesis algorithms are governed by a set of parameters and produce different results depending on which parameter settings are chosen in conjunction with an exemplar used as a basis for synthesis. So far, automatically selecting parameters suitable for synthesis has been a relatively unexplored topic. In effect, this makes texture synthesis supervised rather than fully automatic.

In this technical paper, we propose automatic parameter optimization methods for example based texture synthesis. We cover research to directly estimate specific texture synthesis parameters, such as patch size and iteration convergence, based on input textures. We also examine various similarity measures and evaluate their effectiveness. The goal for each measure is to properly evaluate how well the resulting synthesis compares to the original input.

A good similarity measure will enable the search for the optimal texture synthesis parameters by maximizing the quality of the synthesis as a function of parameters.

We apply presented methods to a state of the art texture synthesis algorithm, namely the one proposed by Kopf et al [62]. It is easy to find a set of exemplars for which there is no single optimal set of settings. The results show a promising foundation for further research in establishing an automated optimal synthesis for a multitude of textures.

9.1 Introduction

Textures are commonly used in computer graphics to enhance the appearance of a scene. Despite the abundance of online texture repositories [1–6], the acquisition of new textures still poses challenges. Example-based Texture Synthesis mitigates this issue by artificially creating new textures from a small input example.

Like most other texture synthesis algorithms, example-based texture synthesis requires manual parameter tweaking to obtain the optimal result in the shortest amount of time. The proper settings aid the algorithms in detecting and recreating the structure present in the examples provided.

We examine two general approaches to improving a state of the art texture synthesis algorithm presented by Kwatra et al. [66] and further refined by Kopf et al. [62]. The texture synthesis algorithm works by minimizing an energy function describing the difference between the input texture(s) (exemplars) provided by the user, and the texture being synthesized. This is effectively done by finding matching texture patches (neighborhoods) and iteratively altering the synthesized patches to look more like the input exemplar(s).

In our first approach we examine methods with which to automatically estimate the optimal synthesis parameters by examining the input exemplar. In the second, we evaluate methods with which to provide a qualitative measure for the resulting synthesized texture. A reliable qualitative measurement would allow us to maximize the quality of the synthesis as a function of the input parameters. We propose a heuristic with which to automate the tweaking of the synthesis parameters based on the qualitative measurement.

The structure of this report is as follows: Related work to both presented approaches is discussed in section 9.2. The texture synthesis algorithm these ap-

proaches are applied to, as well as its associated parameters, is detailed in section 9.3 and 9.3.1, respectively. Work involving the direct estimation of parameters is presented in section 9.4, while section 9.5 covers the indirect estimation of parameters via a qualitative measure. Preliminary results of the direct method is presented in section 9.4, while more comprehensive results from the indirect method is detailed in section 9.6. Current limitations of both approaches is presented in 9.7. We conclude our findings in section 9.8 and discuss future work in section 9.9.

9.2 Related Work

We first present work related to the texture synthesis algorithm to which we apply automated parameter selection. We then detail work specifically related to the direct and indirect parameter optimization approaches.

Additionally, a number of papers exist that present altered and enhanced versions of existing texture synthesis methods, for the purposes of accelerating them. Although this is somewhat removed from the topic of automatic parameter selection, it shares a similar goal. The accceleration of the texture synthesis algorithm while attempting to maintain a quality result. We note a few publications presenting the aforementioned type of research.

9.2.1 Texture Synthesis

A large body of work within texture synthesis research [125] has led to the algorithm presented by Kopf et al. [62].

Texture synthesis algorithms have evolved over the past decade from being parametric [52] to non-parametric [27], pixel [127] and patch-based [67], to optimization-based methods [62, 66]. As previously noted, the publications have focused on either presenting a new and different approach, or evolving an existing method to produce better results. A recent publication [124] has even focused on reverse texture synthesis, which compacts an existing texture down to a smaller representation, from which a new texture is more easily synthesized.

We present results of automatic parameter selection conducted on the optimization-based approach described by Wexler et al. [128], applied by Kwatra et al. [66] and further refined by Kopf et al. [62].

9.2.2 Direct Parameter Optimization

To the best of our knowledge, no paper exists with the explicit goal of optimizing the given parameters of a texture synthesis algorithm, apart from the paper presenting the algorithm itself or iterative work upon the same. This is not especially surprising, given that the type of parameters eligible for optimization depend entirely on the synthesis algorithm itself. In this report, we focus on examining exemplar based texture optimization.

A parameter suitable for direct optimization is the size of the aforementioned neighborhoods used during synthesis. Section 9.4 provides a more detailed explanation to this effect. Briefly, the optimal neighborhood size is the smallest possible size, while still encompassing all unique structures captured in a texture. Hong et al. present a novel method with which to estimate texture scale [54] and apply it to set of highly periodic brodatz textures [15].

9.2.3 Indirect Parameter Optimization

Similar to direct parameter optimization, to the best of our knowledge, no paper exists that specifically investigates the impact of varying parameters used during texture synthesis, for the purposes of making the algorithm fully automatic. However, within the broader spectrum of general computer science research, automatic parameter tuning based on an algorithms final result is common.

The automatic tuning of parameters with regards to a quality measure is sometimes called a metaheuristic, and is a subfield of stochastic optimization. A large body of work exists within this field dating back to the early 1950s. Luke and Talbi each provide a perspective over the current state of these types of algorithms as well as implementation based examples [80, 113]. Nanono et al. provide multiple concrete applications of parameter tuning in modern computer science problems [90].

9.2.4 Accelerating Texture Synthesis

Because texture synthesis is a computationally demanding task, it is only natural that research into improving performance or alleviating the calculatory burden exists. Lefebvre and Hoppe [77] extend Wei and Levoy's 2D synthesis approach [126] by parallelizing it and implementing it on modern GPU hardware.

Manke and Wünsche [83] extend Lefebvre and Hoppes approach allowing it to synthesize solid textures while executing on a modern GPU. Their paper provides a thorough explanation and a speculative GPU performance forecast, but their implementation is limited to a software prototype running in `C++`.

Dong et al. [30] present a novel method that synthesizes solid textures to cover the surface region of a given mesh. Their approach yields impressive results at high speeds, but due to their reliance on precomputed seamlessly interconnected neighborhoods, the algorithm can introduce a bias during synthesis, eliminating potentially significant features.

Recently, Barnes et al. have presented an algorithm that significantly increases the speed of finding the best approximate match for a patch in a given texture [11].

9.3 Texture Optimization

The synthesis algorithm we test our methods on was originally presented in Kopf et al.'s paper [62], and more thoroughly detailed in a related research paper [72]. In this paper we will restrict our explanation of the algorithm to the portions where we deviate from the aforementioned descriptions, as well as portions related to the synthesis parameters which we tweak and analyze.

In short, the texture optimization algorithm attempts to minimize an energy function describing the difference between the input exemplar and the texture being synthesized. A simplified version of the function detailed by Kopf et al. is

$$E(N_s, N_e) = \sum_{i=1}^{n_s} \|N_{s,i} - N_{e,best}\|^r. \tag{9.1}$$

The input parameters (N_s) and (N_e) represent the texture being synthesized and the input exemplar respectively. To measure the energy difference between N_s and N_e, small texture patches (usually 8 by 8 pixels) are extracted and compared. The total number of patches (a.k.a neighborhoods) from N_s is denoted n_s. For each of the neighborhoods extracted from the synthesized texture $N_{s,i}$, its corresponding best match (measured via L_2 norm distance) is subtracted ($N_{e,best}$). The sum of differences describe the energy difference between the two textures. Setting the exponent $r = 0.8$ in the energy function keeps it more robust against outliers [62, 66].

The synthesis algorithm progresses through several levels of detail, starting with

a coarse 32x32 resolution synthesis texture, comprised of random samples from the input exemplar. For each extracted synthesis neighborhood, the approximate best matching neighborhood is found. Finally, every pixel is updated based on those best matching neighborhoods. The process is comparable to an expectation maximization algorithm. The best matches are found, the whole texture is improved, and finally the process repeats itself.

The neighborhoods extracted from the synthesized texture lie on a sparse grid spaced 2 pixels apart as shown in Figure 9.1, where as neighborhoods extracted from an exemplar lie on a densely populated grid. In the dense grid, each pixel can be thought of as representing a single neighborhood.

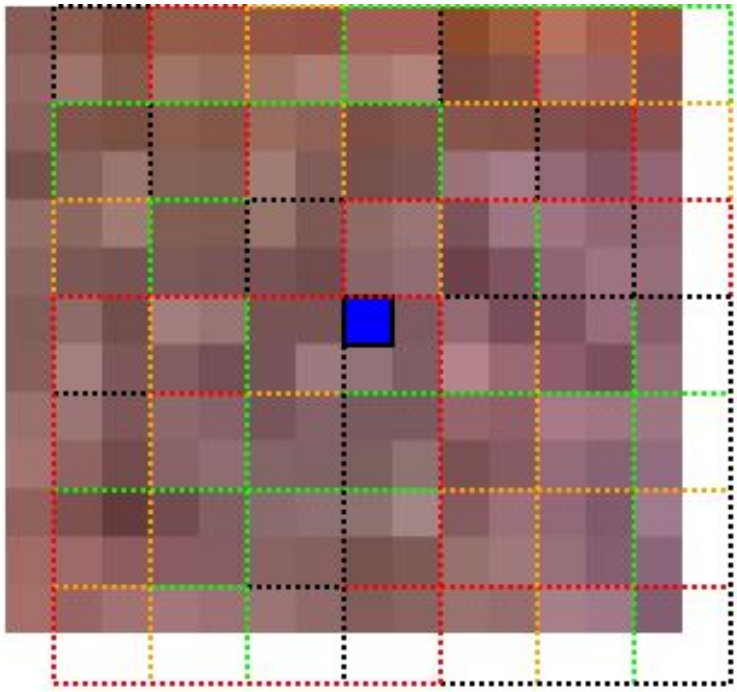

Figure 9.1: Density of extracted neighborhoods in the synthesis texture. Visualized are all the 8x8 neighborhoods which the blue highlighted pixel is a member of.

Since the exemplar used to synthesize a new texture usually contains color, each vectorized neighborhood is comprised of 192 values, consisting of three color channels for each of the 8x8 pixels in the neighborhood. Finding the best matching exemplar neighborhood for each synthesized neighborhood is a computationally expensive task in such a high dimensional room, so prior to finding matches, the dimensionality of the neighborhoods is reduced using principal component analysis (PCA) to a state where 95% of their variance is still retained

($\sigma = 0.95$). To further increase the speed of searching for each neighborhoods nearest neighbor, an approximate nearest neighbor algorithm is utilized from the ANN Library [89]. It requires a distance parameter ε to be set, which is usually set to 2 [62], guarenteeing that all found neighborhoods lie no further than $1 + \varepsilon$, times the optimal match distance, away.

Once the approximate best neighborhoods have been located, the algorithm employs a clustering approach proposed by Wexler et al. [128] to reduce the number of contributing neighborhoods to a single pixel. This speeds up convergence by removing outliers and constrains the neighborhoods, that each pixel is a part of, to a group that contribute a similar color.

Additionally, the texture optimization approach employs a histogram weighing scheme that prunes contributions which overshoot the current amount of color, in the respective channel, for the pixel being updated. This ensures overall global correspondence to the input exemplar, while the neighborhood matching serves to increase local spatial correspondence.

In some cases it is beneficial, and occasionally a requirement, to attach a feature map (as an additional channel) in order to produce a satisfactory result, when synthesizing a troublesome texture. The feature map basically assigns a weight to each pixel, indicating its importance with regards to the textures structure. This weight can be adjusted as necessary, if the algorithm is having trouble reproducing the pattern found within the texture.

9.3.1 Synthesis Parameters

Texture optimization provides multiple adjustable parameters. This section reiterates the ones we believe are most significant, along with a brief description. How each parameter is affected during direct and indirect optimization is explained in sections 9.4 and 9.5, respectively.

- **Neighborhood size** — The size of the texture patches compared in between the input exemplars and the texture being synthesized. Kopf et al. define the default Neighborhood size as 8 by 8 pixel.
- **Neighborhood grid density/sparsity** — The density/sparsity of neighborhoods extracted from the exemplar and synthesis textures. Kopf et al. extract neighborhoods from a dense grid on exemplars, and from a sparser grid (2 pixels apart) on the synthesized texture.
- **Neighborhood dimension reduction (σ)** — The method and severity

with which the dimensionality, of the vectorized neighborhoods, is reduced. In the method proposed by Kopf et al., the variance is reduced to 95% by using PCA.

- **Convergence** — The method with which convergence of the synthesized texture is determined. Kopf et al. use a set number of iterations for each level of detail (Johannes Kopf, personal communication, March 10, 2010).
- **ANN Distance (ε)** — A parameter defining that the best matching exemplar neighborhood, found for a given synthesized neighborhood, is guaranteed to be no further than $1 + \varepsilon$ times the distance to the actual closest exemplar neighborhood.
- **Histogram matching weight adjustment** — When a new color is determined for a given synthesis texture pixel, each contributing color is compared with the already existing color contribution in the whole synthesis texture. If the amount of color in the synthesized texture is higher than that of the exemplar texture, then the contribution is punished as detailed by Kopf et al. [62]. This value is further amplified by a static weight parameter. Setting this parameter value to zero will nullify the effects of histogram matching completely.
- **Clustering algorithm** — The clustering algorithm intended to accelerate convergence and remove outlying contributors. A meanshift algorithm is employed by Kopf et al. with a number of threshold parameters.
- **Feature map channel** — The feature map represents an individual weight parameter for each pixel (or voxel) in the input exemplar(s). The static weight associated with the feature map can be adjusted as needed for the synthesis algorithm to converge successfully.

9.4 Direct Parameter Selection

Most of the parameters associated with the texture optimization algorithm presented by Kopf et al. [62] affect unique portions of the algorithm itself. We apply methods specifically aimed at automatically selecting the individual parameters presented below.

- **Neighborhood size** — Kopf et al. [62] note that the use of histogram matching allowed for the use of small neighborhoods (`8x8`), while still recreating the features of the original input exemplar. While this is true, our empirical testing revealed improved results with certain 2D textures using a larger neighborhood size, as visualized in figure 9.10.

The optimal neighborhood size is undoubtedly dependent on the textures used as input. Recreating the features found in these textures, is a question of scale. A larger neighborhood is more suitable to properly recreate the features of a texture with a larger scale, where as a smaller neighborhood is suitable to a texture with a smaller scale, as visualized in figure 9.2.

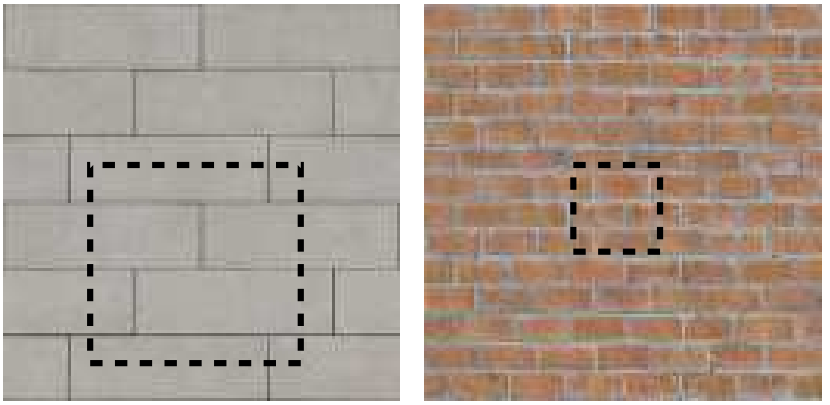

Figure 9.2: An approximation of the ideal neighborhood size given two differently scaled textures. On the left, the brick wall with the much bigger scale

- **Convergence** — As previously mentioned, Kopf et al. use a set number of iterations for each level of detail (Johannes Kopf, personal communication, March 10, 2010).

  Through empirical testing, we have determined that certain texture will converge much faster than others during synthesis. An approach measuring convergence would avoid wasting computational power by stopping the synthesis after an acceptable result has been achieved.

Optimization of the remaining parameters is beyond the scope of this report.

9.4.1 Neighborhood size estimation

As previously mentioned in section 9.2, Hong et al. [54] present a novel method with which to estimate texture scale, using the following scale descriptor applied on each neighborhood N_e:

$$\inf_r D(N_{e,r}, N_{en,r}) - \alpha H(N_{e,r}) + \beta r(x). \tag{9.2}$$

Texture	Mean	Median
Brodatz (a)	22.3423	21
Brick wall (b)	17.6359	15
Zebra stripe (c)	24.3031	27
Animal Skin (d)	26.5022	27
Big Brick (e)	10.7212	9
Small Brick (f)	17.8322	27

Table 9.1: The mean and median of the scales estimated for every pixel in figure 9.3.

The equation minimizes the energy measured by three terms, using the variables $N_{e,r}$, $N_{en,r}$, and r. The current neighborhood with a "radius" r (height and width), is denoted $N_{e,r}$. It's surrounding neighbors is denoted $N_{en,r}$. The first term measures differences between the two regions either via Kullback-Leibler or Wasserstein distance. In this report we focus on the Wasserstein distance, as it yielded more reliable results during testing. The second term compensates for comparing homogenous patches, by reducing the energy proportionally to the amount of entropy measured in the neighborhood patches. Finally, the last term ensures that the equation favors patches with as small a size as possible. The weight parameters α and β are set to 0.001 and 0.1, respectively, as suggested by Hong et al. [54].

We applied Hong et al.s method to one of the brodatz textures [15], as well as a number of textures used by Kopf et al. [62], visualized in figure 9.3. Table 9.1 display the measured mean and median for each of the textures.

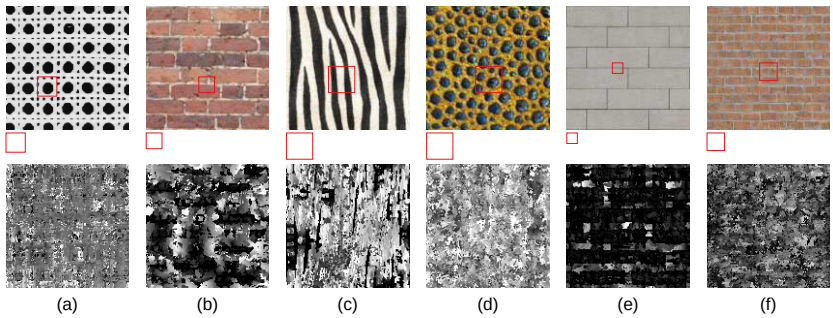

Figure 9.3: The top row shows the original textures, along with the median neighborhood estimated from every single pixel as a red box in the texture. The same neighborhood size is also visualized immediately below the texture. The bottom row is a scale map representation of each of the textures, obtained by applying Hong et al.s energy equation 9.2, where each pixel is given an intensity matching the estimated best neighborhood size surrounding that pixel. The more intense the pixel, the larger the estimated neighborhood for that location.

The results do not always correlate directly with the size of the structural elements in the textures. The most striking example of this is the estimation of scale for the big bricks (e) and the smaller bricks (f). This is likely the cause of the homogeneous nature of the large gray areas contained within the brick texture which lead to a high similarity in the first term of equation 9.2.

Although the method delivers some results similar to our own estimation of optimal neighborhood size, there are also notable failure cases, such as the big bricks (e) and the smaller bricks (f).

We consider two alternative methods of estimating texture scale, based on an analysis of the relationship in between the neighborhoods:

9.4.1.1 Neighborhood Clustering

Texture Optimization [62] uses clustering as a means to speed up convergence, by discarding contributions which are not a part of the dominant cluster while updating a single pixel. As previously shown in figure 9.1, several neighborhoods contribute to a single pixel, and mean-shift clustering ensures that only contributions from the main cluster is retained.

This type of clustering approach could also be used as a scale descriptor. By expressing each neighborhood of a texture as a point in a high dimensional space, it may be possible to estimate scale based on this distribution. For each neighborhood size, meanshift clustering is applied to determine the dominant cluster. The optimal neighborhood size would have the biggest cluster.

There are issues that require further attention while implementing this approach:

- **Neighborhood Size** — As the size of the neighborhoods increase in an attempt to find the biggest cluster, so does the number of dimensions that each neighborhood "point" lies on. Principal component analysis is a useful tool in both reducing calculatory complexity as well as limiting the number of dimensions for each "point". However, it would still be necessary to include a weight parameter for the purposes of counter balancing this increased difficulty in clustering due to the higher number of dimensions.
- **Meanshift Threshold** — The meanshift clustering algorithms works with thresholding values that would need to be adjusted empirically to determine an optimal setting for the majority of textures. Since the point is to automatically estimate parameters, and not replace these with other

parameters, the thresholds should either be established automatically or perform well for all textures of a given type.

- **Mahalanobis distance** — The distance between neighborhoods in the high dimensional space could potentially be better estimated using the mahalanobis distance, instead of the euclidean distance.

Neighborhood Tree Structure An alternate approach of examining the relationship in between neighborhoods is by building a tree structure representing nearest neighbors. Starting with a random neighborhood from the input exemplar, we form a new cluster consisting of that one member. Each cluster is represented by a single neighborhood, i.e. an average of all the existing neighborhoods in that cluster. We then continuously add the remaining neighborhoods to the tree structure. If the new neighborhood is within a certain threshold distance of the representative neighborhood of a cluster, it is added to the same cluster. Otherwise, it will form its own new unique cluster.

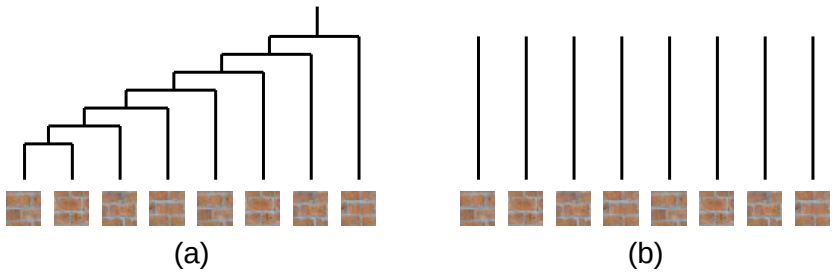

Figure 9.4: (a) The ideal tree structure where each new neighborhood is added to the already existing cluster. (b) The worst case scenario where each neighborhood forms its own cluster.

Figure 9.4 shows both the ideal, and worst case scenario of the tree structure. As with the clustering method, there are a few issues that require further attention:

- **Cluster Representative Neighborhood** — The best method of representing a cluster should be further investigated. One option would be to pick the neighborhood closest to all other neighborhoods in the entire cluster. An alternate option would be to simply average all the neighborhoods in the cluster together and use that as a representative. Although the latter option would likely cause unwanted blurring and should be carefully considered.
- **Distance Threshold** — The optimal threshold for determining if a neighborhood is close enough to a cluster to become a member should be em-

pirically tested.

- **Mahalanobis distance** — Just like with the neighborhood clustering algorithm, the mahalanobis distance could also be applied here when introducing new neighborhoods to existing clusters.

9.4.2 Convergence estimation

As previously noted, Kopf et al. rely on a fixed set of iterations for each level of detail, during texture synthesis. We experimented with both the L_1- and L_2-Norm applied in conjunction with the energy function (in equation 9.1), as a useful measurement for synthesis convergence.

We were unable to achieve acceptable results using a fixed and/or dynamic threshold. A solution might be using a probabilistic metaheuristic to provide a good approximation to the global optimum, such as simulated annealing [131].

9.5 Indirect Parameter Selection

The core of our indirect parameter optimization method is straightforward. We attempt to establish an objective measurement of texture quality. Assuming that this objective measurements correctly identifies the best synthesized texture among several candidates, then determining the optimal parameters for a specific texture can be solved using a pure brute force method. If we also assume that the parameters are at least moderately orthogonal, we can approach the problem in a linear fashion. By only varying a single parameter, we can determine its optimal setting. Applying our objective measurements on each synthesized result while varying one parameter, we determine which setting produces the best result, for that parameter. We continue until we've determined the optimal setting for each parameter.

We break down the indirect method into five separate steps and detail in each how we mitigate the complexity arguably without reducing the quality of the results. For the benefit of the reader, we list the steps in an abbreviated form below:

1. Select parameters to optimize
2. Select similarity measures

3. Determine parameter bounds using similarity measures
4. Evaluate results of varying parameters within bounds using similarity measures
5. Perform automated texture synthesis

These steps are thoroughly detailed in identically ordered subsections below.

9.5.1 Tested Synthesis Parameters

Ideally we would like to measure the effects on all parameters involved in the texture optimization process. However, certain parameters are arguably harder to optimize, and the complexity of determining optimal parameters can increase exponentially the more parameters are involved. Below we detail which parameters we vary, and which remain static.

1. Fixed
    - **Neighborhood Grid Density/Sparsity** — Although we firmly believe that certain textures could easily produce an acceptable result with a much sparser set of neighborhoods, we've chosen to constrain the complexity of our analysis, and keep the recommended neighborhood sparsity on the synthesized texture as recommended by Kopf et al.
    - **Convergence** — Similar to Kopf et al. we rely on a fixed set of iterations per detail scale to achieve a successful synthesis result. We found that the algorithm almost always converged when using 100, 30, and 10 iterations for the 32, 64, and 128 pixel resolution scale respectively.
    - **Feature map** — Originally introduced by Wu and Yu [132], feature maps aid patch-based texture synthesis methods in reproducing the structure found in the original input exemplar. All original feature maps require user input and are created artificially. We've chosen to focus on textures that do not require feature maps in order to produce acceptable results.
    - **Clustering Algorithm and associated Parameters** - Kopf et al. note in their paper [62] that purely averaging all the contributing colors for a single pixel might produce blurry results. While this sometimes occurred during our testing procedure, it is our impression

that blurring, during 2D synthesis, only happened when there was no way for the algorithm to satisfactorily converge, in a particular region. Meanshift clustering on the other hand would force convergence and cause an unsightly seam to appear in its place.

However, we did find that if histogram matching was not applied, simply averaging all contributors would occasionally cause the synthesis algorithm to yield unsatisfactory results.

We also applied K-Means as an alternate clustering algorithm, and found that it produced results slightly less favorable when compared to Meanshift clustering, but had a much faster runtime. Almost comparable to simple averaging. Since the best textures were achieved using straight averaging of all contributors in conjunction with histogram matching, we utilize it exclusively during our testing procedure.

- **Histogram Weight Adjustment** — Originally intended as a non-static parameter, testing revealed that the best results were consistently achieved with a fixed value. Since there is no computational gain from varying the parameter (except for turning it off), we're keeping it static. Table 9.2 notes which settings were tested, as well as the permanent setting chosen.

2. Variable

- **Neighborhood Size** — Kopf et al. suggest using 8 by 8 pixel sized neighborhoods during the synthesis process. While determining the upper and lower bounds for this parameter we found this neighborhood size to often be the threshold for where a number of textures started converging properly.
- **Neighborhood Dimension Reduction (σ)** — Retaining 95% of the original textures variance is suggested by Kopf et al. and works well for all tested textures. A further reduction in retained variance sometimes shows little to no visual artifacts, where as other textures will immediatly cease to produce an acceptable result.
- **ANN Distance (ε)** — Kopf et al. suggests setting $\varepsilon = 2.0$ during synthesis. A higher setting leads to less exact/faster neighborhood matching, whereas a lower setting will generally be slower and produce more exact matches. A side-effect of a low setting (2.0) is that the algorithm occasionally synthesizes a near exact replica of the input exemplar, albeit with a vertical and horizontal offset. An example of this can be seen in Figure 9.5.

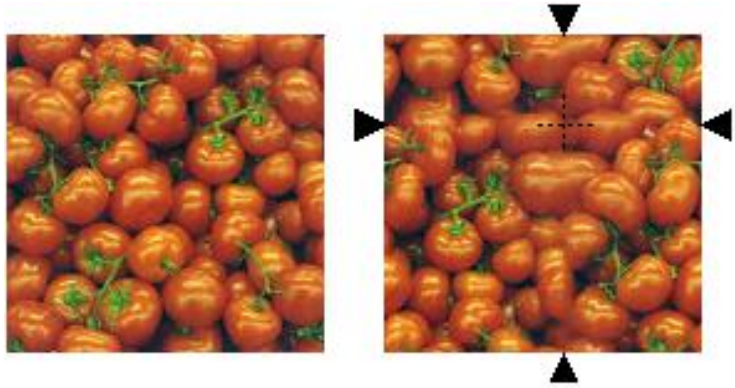

Figure 9.5: On the left, the original tomato exemplar used as input. On the right, is the resulting synthesized 2D texture using the parameters as suggested by Kopf et al. Notice that the original exemplar has been reproduced in its entirety with the original edges pointed out by the arrows.

9.5.2 Texture Similarity Measurements

Determining the quality of a synthesized texture might seem trivial at first, since we are so used to comparing and evaluating what we see as humans. In fact, stating that people are living, breathing pattern recognition machines wouldn't be far from the truth. But in addition to a subjective evaluation, we are interested in obtaining quantifiable objective measurements. Below we detail which similarity measures we test to help us determine how successful a synthesized texture is in recapturing the original textures variety and likeness.

- **Subjective comparison** — While objective measurements aim to automatically quantify our synthesized results, they cannot capture the human impression given by the resulting texture. What we perceive remains a cornerstone of graphics development and as such, the subjective impression cannot be disregarded. We therefore perform a subjective evaluation of the results that the synthesis process yields, in addition to the objective measurements.
- **Reverse neighborhood look-up comparison** — During the synthesis process, the approximate best matching exemplar neighborhood is found for each synthesized neighborhood. Treating each neighborhood as a 192 dimensional vector and calculating the distance using the L_2 Norm yields an objective measurement of how much the textures differ within that neighborhood region. It seems like an ideal method when applied to all neighborhoods in order to get a sense of how well the resulting synthesized

texture turned out.

Unfortunately, this is not the case. Matching each synthesized neighborhoods to its best matching exemplar, as done during texture synthesis, can yield a result indicating high similarity, even if the whole synthesized texture only resembles a small portion from the input exemplar. Since we want to punish textures for not making full use of the variance provided by the input exemplar, we instead match each exemplar neighborhood to it's best matching synthesized neighborhood on a dense grid and calculate the L_2 as follows

$$Diff(N_e, N_s) = \frac{\sum_{i=1}^{n_e}(N_{e,i} - N_{s,best})^2}{n_e}. \tag{9.3}$$

The total number of neighborhoods derived from the exemplar texture is denoted n_e and a single exemplar neighborhood and its best matching synthesized neighborhood is denoted by $N_{e,i}$ and $N_{s,best}$ respectively. Note the differences between this equation (9.3), and the equation used during synthesis (9.1):

- For each extracted **exemplar** neighborhood, the best **synthesized** neighborhood is found.
- The actual best match is located during comparison (i.e. $\varepsilon = 0$).
- No dimensionality reduction is performed (i.e $\sigma = 1.0$).
- Neighborhoods are sized `10x10`, and are extracted from a dense grid on **both** the synthesized and exemplar texture.

Texture optimization actively minimizes the difference between synthesis and exemplar neighborhoods, so there exists a direct coupling between the process of obtaining a synthesized texture and this particular measurement of its objective quality. We perform the comparison using `10x10` sized neighborhoods, as opposed to the default `8x8` setting as suggested by Kopf et al.

Each channel is scaled to fit between 0..1, and the end result is divided by the number of neighborhoods extracted from the exemplar. The potential range for this objective measurement therefore spans between 0 and 192.

This approach is similar to the bidirectional similarity measure presented by Simakov et al. [108].

- **Crude Reverse neighborhood look-up comparison** — The *reverse neighborhood look-up comparison* test (*rnlc* test) is computationally expensive. It sacrifices no quality, and will give the best indication of whether the test itself produces viable results. This crude version of the same test sacrifices some quality in the hopes of achieving comparable results to the

rnlc test, at much faster speeds. Compared to the *rnlc* test it retains 95% variance (as opposed to 100%), and locates approximate best matches ($\varepsilon = 2.0$), similar to the settings used during synthesis.

- **Global histogram difference** — To ensure a global texture correspondence, we calculate the histogram difference between the input exemplar and the synthesized result via the following equation:

$$HistDiff(T_e, T_s) = \sum_{c=1}^{j} \sum_{b=1}^{k} \| \frac{T_{e,b,c}}{v_e} - \frac{T_{s,b,c}}{v_s} \|. \tag{9.4}$$

The variable j denotes the number of channels being synthesized, as determined by the input exemplar. All of our exemplars are RGB colored and therefore $j = 3$. The total number of contributing pixels from the exemplar and synthesized texture is denoted by v_e and v_s respectively. The number of bins used to compare histograms is denoted k. This similarity measure, like the *reverse neighborhood look-up comparison*, is coupled to the synthesis process. Contributing pixels that do not conform to the exemplars histogram are pruned during synthesis. To minimize the coupling we set k to a maximum of 255 bins, instead of 16 used during actual synthesis.

If the histograms from the exemplar and the synthesized result are identical, eqn. 9.4 will yield a zero sum. The most opposite textures (a white and a black one) would yield a maximum of $j * 2$ in difference.

9.5.3 Determining parameter bounds

As previously mentioned, we attempt to solve the problem of optimizing our parameters using linear brute force. The approach is simple, yet functional. However, it is far too computationally expensive to be viable. There are several parameters involved and some of them have potentially infinite settings. By only varying our parameters within fixed upper and lower bounds, we can optimize the approach arguably without losing any significant resulting quality.

When reducing our parameter search space, we use an extended set of textures, shown in Figure 9.7. Most textures are from the same set which Kopf et al. [62] used to produce solid textures with. The remaining few textures originate from the CGTexture repository [1].

We synthesize results for each permutation of parameter settings to visually determine the bounds for the parameters (detailed in Section 9.5.1). In other

words, a single test consists of fixing every parameter to the defaults as defined by Kopf et al. and varying one parameter within a heuristically determined range. To avoid outliers we synthesize a minimum of four textures and visually compare these to the original input texture.

A number of parameters have a natural upper bound where neither quality nor precision is sacrificed, at the expense of computational complexity. For example, setting the ANN distance (ε) to 0 ensures that the best matching neighborhood is always found, or keeping 100% of the variance (σ) during the principle component analysis leads to no reduction in quality. We define this as the upper bound for these types of parameters. The lower bound is found by incrementally sacrificing more and more quality until the synthesis algorithm only produces indiscriminant noise for the majority of test textures. Figure 9.6 shows the synthesis results for three different textures retaining a continously lower amount of variance, while applying PCA. Note that while all textures degrade in quality, some degrade much faster than others depending on the complexity of structure found in the original texture.

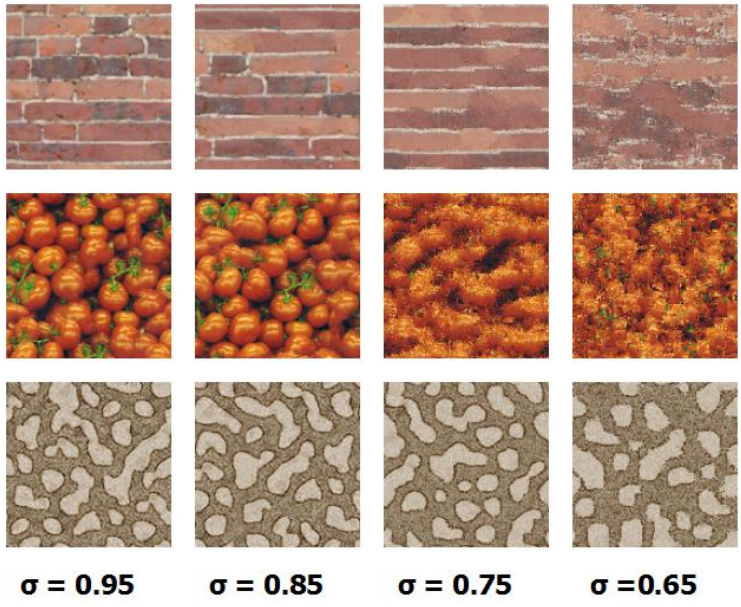

Figure 9.6: Synthesized results using three different exemplars. From left to right, each column shows the end result produced while retaining less variance (σ).

Figure 9.7: The fourteen different input exemplars we use to determine the various bounds of the synthesis parameters. The blue highlighted exemplars on the left are also used to measure the efficiency of our objective measurements.

9.5.4 Evaluate objective measurement

Once the upper and lower and lower bounds have been determined, we continue our testing with a smaller set of textures (shown in Figure 9.7), due to computational complexity. For each of the four textures, we synthesize a set of 10 textures for each permutation of settings within the bounds determined by the previous tests. During these tests we apply our objective measurements to each of the synthesized results to determine how well they reflect the quality of the texture when compared to the visual impression.

9.5.5 Automated texture synthesis

Finally, we use the objective measurements to perform a small set of automatic synthesis runs using the objective measurements as guide for our algorithm to determine the optimal setting for each parameter.

9.6 Indirect Parameter Selection Results

Having selected parameters and similarity measures as detailed in Section 9.5, we visually determined the upper and lower bounds as listed in Table 9.2, using the extended set of textures shown in Figure 9.7.

After determining these bounds, we proceed to test our similarity measures as explained in Section 9.5.4. The similarity measures are tested by completing a total of 10 synthesized textures per settings permutation using a reduced set of textures, as shown in Figure 9.7. The results indicate that the *global histogram*

Parameter	Preliminary Test Settings	Upper bound	Lower bound
NB Size	2x2, 3x3, ... 15x15, 16x16, 20x20, 30x30	16x16	5x5
Dimension Reduction (σ)	0.95, 0.90, ... 0.70, 0.65	0.95	0.75
ANN Distance (ε)	1.5, 2, 3, 4, 8, 16, 28	1.5	12
Hist Punishment	0, 0.5, 1, 2, 4, 8, 16, 32, 64, 128, 1000	128.0	128.0

Table 9.2: The parameters and their settings used during preliminary testing to determine upper (best quality) and lower bounds (worst quality).

difference measurement does not correlate well with our visual impression of the quality. It's not entirely surprising given that it measures global color correspondence and not structural similarity. A concrete example of lacking visual correlation is shown in Figure 9.8. Using small neighborhoods (2x2 and 3x3) during synthesis measure as being better than using larger neighborhoods (4x4, 5x5 and 6x6), when using *global histogram difference* as a similarity measure. It's clear that its due to the green tomato stalks missing from these results. Using bigger neighborhoods, they reappear and the *global histogram difference* measurement confirms that these results are better.

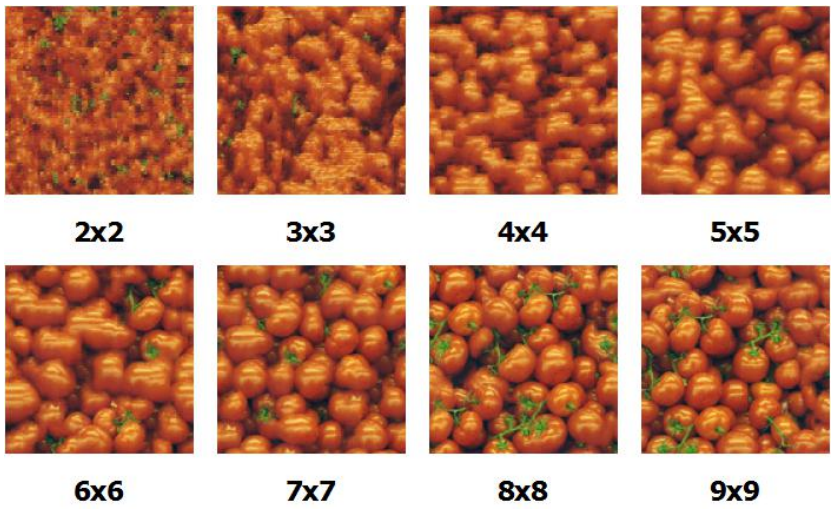

Figure 9.8: A synthesized result for different sizes of neighborhoods, using the tomato picture as input exemplar.

Consequently, the *global histogram difference* measurement cannot be used to

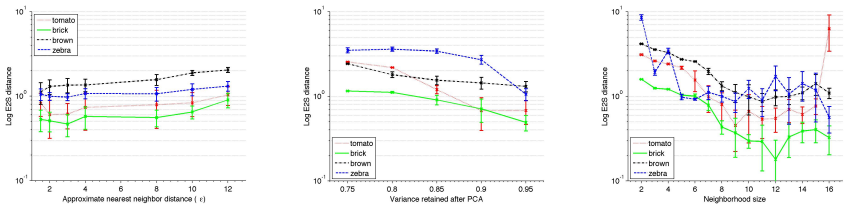

Figure 9.9: Three plots showing the results of the reverse neighborhood look-up comparison while varying the parameters specified in Section 9.5.1. Each plot shows the average result of 10 tests including std. deviation. The line colors refer to which texture being synthesized, as detailed in the legend.

singlehandedly determine if a result turns out well or not. The *Reverse neighborhood look-up comparison* on the other hand correlates well with our visual impression of the synthesized results. Figure 9.9 shows a general tendency of an improved result as the approximate nearest neighbor distance is reduced, or a higher variance is retained. It's worth noting that the brown dirt texture is the least affected by a reduction in variance, which the results correlate with. Figure 9.10 shows the results of applying the optimally detected neighborhood size along with the default $\varepsilon = 2.0$ and $\sigma = 0.95$ settings.

The *rnlc* test takes near a minute to complete with a mean completion time of 42.8 seconds and a std. deviation of 26.6 seconds. Fortunately, we found that the crude *rnlc* test performs equally well when looking at each texture individually. Specifically, the objectively measured results closely resemble those of the regular *rnlc* test, except for being consistently better or worse. The mean completion time for the crude *rnlc* test is 4.6 seconds with a std. deviation of 1.9 seconds. Approximately 10 times faster.

Treating each parameter as orthogonal and performing a linear search through the parameter settings space occasionally causes the algorithm to select parameters which are not ideal, resulting in a sub optimal synthesized texture. To mitigate this issue, we've chosen an order by which the parameters are tuned, and once an optimal value for a parameter is found, we retain it when tweaking the remaining parameter. Figure 9.11 shows a comparison between a standard and optimized synthesis result using the four highlighted textures from Figure 9.7 as input.

The runtime of automatically tuning the parameters is highly dependent on the complexity of the texture used as an input exemplar. Using the four textures shown in Figure 9.11, the whole process took process, including synthesizing the final result, lasted between one and four hours, depending on the texture.

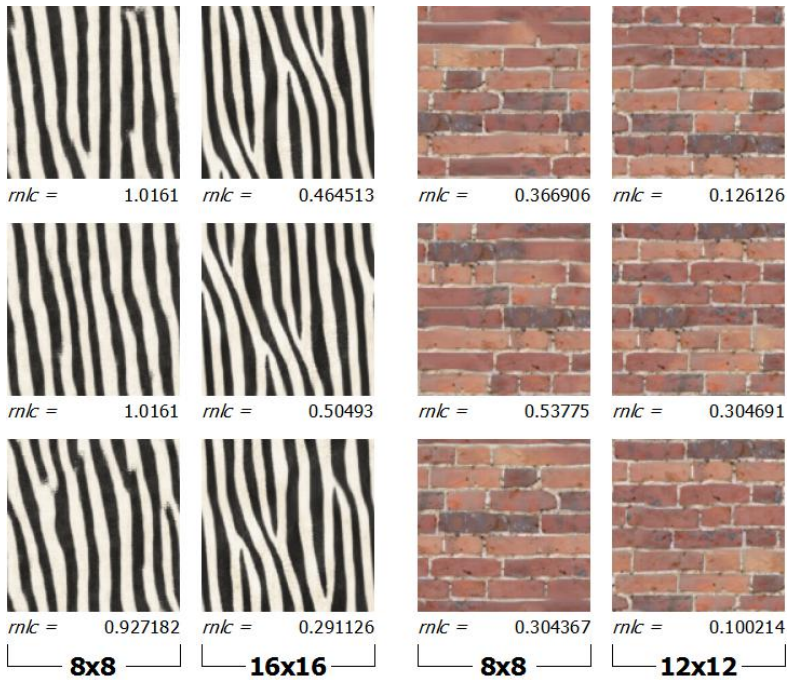

Figure 9.10: Synthesized results using the zebra (left) and brick (right) textures as input exemplars. The first/third column show three different results of using the parameters suggested by Kopf et al. (neighborhoods sized `8x8`) and the second/fourth column shows the results using the objectively measured optimal neighborhood sizes (`16x16` and `12x12`). Below each result is the qualitative measurement calculated via the *rnlc* test.

9.6.1 Summary

The extensive testing described in the previous sections have resulted in the development of the following heuristic, for automatically adjusting parameters of example-based texture synthesis algorithms.

1. Isolate parameters intended for optimization, and use the crude *Reverse neighborhood look-up comparison* test, as described in Section 9.5.2, on a wide range of settings to determine the upper and lower bounds for each. In the case of texture optimization, the determined detailed in Table 9.2 work well.

2. Apply a metaheuristic, like linear search, and determine the optimal setting using at least 10 measurements via the crude *rnlc* test. Optimize

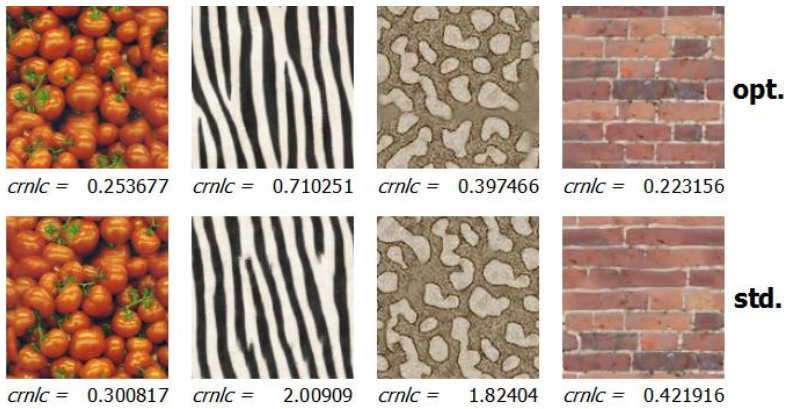

Figure 9.11: Comparison of the four sample textures synthesized using automatically detected optimal settings (top row) and standardized settings used by Kopf et al. (bottom row). The quality as measured by the crude *rnlc* test is listed below each result.

parameters in the order of approximate nearest neighbor distance, neighborhood size, and PCA variance retention. Once an optimal parameter has been determined, retain it while optimizing the remaining parameters.

3. Synthesized texture using the determined optimal settings.

9.7 Limitations

Automatic parameter selection is no small task, and in some ways, this technical report only scratches the surface. The following section sums up the achievements of our efforts, while this section details short-comings we have identified with either method.

9.7.1 Direct Parameter Selection

Preliminary findings using direct parameter optimization indicate that the work by Hong et al. [54] produces suboptimal results in certain cases. Although the work shows promise, we question whether a probability density distribution is a sufficiently accurate representation of the different neighborhoods when determining scale. All spatial information is discarded in this representation which is limiting.

The alternate methods are in the preliminary stages, and therefore not viable for a proper evaluation of limitations. However, it is clear that both alternatives have the potential to introduce further parameters into the process, which would undo the purpose of this research. It is important that these methods either function well with a set of static parameters, or handle are capable of self-estimating them.

9.7.2 Indirect Parameter Selection

- **Objective Quality Measurement** — The crude *Reverse neighborhood look-up comparison* generally delivers favorable results. There are however some failure cases, such as the brown textured result in figure 9.11 having a much lower score than the standardized version, despite being more blurry. Additionally, the measured quality of a texture is not monotone and contains local minima.

  Using our determined upper and lower bounds lessens the risk of the algorithm falling into a local minima, but does not eliminate it entirely. It is also important to note that multiple measurements are required for the crude *Reverse neighborhood look-up comparison* test to deliver a consistent result. Looking at the results in Figure 9.9, its clear that the measurements can deviate considerably, and to compensate we would recommend measuring the result more than ten times. Since the runtime of the algorithm is already 'high' and further tests would only exacerbate the situation, it would be advantageous to apply a more advanced metaheuristic, such as simulated annealing.

  The high number of measurements required for a reliable estimation, causes the algorithm to run for several hours. Implementing Barnes et al.s algorithm [11] should cause a considerable speed up overall.

  Finally, it would also be advantageous to extend the similarity measure to incorporate detection for visual elements we find unappealing, such as blurring or hard edges. In its current state, the *Reverse neighborhood look-up comparison* would highly favor a result that is strikingly similar to the original input exemplar (such as the result shown in Figure 9.5).

- **Parameter Order Bias** — Since we've selected a fixed order in which we optimize our chosen parameters, as detailed in Section 9.6.1, we introduce a potential bias into the system.

- **Global Histogram Difference Measurement** — This measurement has some short comings. One of the most critical ones being, a case where one neighborhood is just a single shader darker than the neighborhood it is currently being compared to. Currently, the estimate would rate the two

neighborhoods as very different, although they would optically be quite similar.

Reducing the number of bins would alleviate the problem. Another solution would be to apply a histogram less prone to overall shifts in intensity, such as earth movers distance [102].

- **Neighborhood Size** — Our objective measurement uses a specific sized neighborhood and consequently introduces an amount of bias. This bias could be minimized by expanding the objective quality measure to include multiple sized neighborhoods, which may be feasible given the computational optimizations provided by patchmatch [11].

- **Quality Vs. Speed** — Even with automatic parameter selection, there is still an implicit trade off between quality and speed. Allowing the user a single parameter controlling this aspect of the algorithm would be an improvement on the approach.

- **Overall speed** — The proposed method is slow when compared to manual parameter tuning. Apart from using a more advanced metaheuristic to traverse the search space, it might be possible to tune the parameters while producing a smaller, and therefore faster, texture patch (instead of the `128x128` sized result we synthesize currently). The parameters tuned while producing a smaller texture patch could serve as an initial 'good guess'. Reducing the amount of information from the input exemplar could also be achieved by first synthesizing a monochrome version.

9.8 Conclusion

We've presented analyzed and presented both direct and indirect methods for the purposes of automatic texture synthesis parameter selection. The direct methods are advantageous as they, by definition, provide a more direct route to automatically estimating parameters without the need for relying on a similarity measure, which potentially introduces further bias.

The heuristics developed as part of our indirect approach, combined with an objective similarity measure is capable of successfully synthesizing a better result than those using a standard set of parameters. The approach is not strictly limited to texture optimization (as presented in this paper), but could be applied to any exemplar-based texture synthesis algorithm.

Neither of the methods are currently without flaws. We identify the most notable limitations and note upon potential solutions. We believe that methods shows

promise as a viable method for fully automating texture synthesis, and warrant further research.

9.9 Future Work

All of the limitations detailed in section 9.7 are ideal for future improvement, especially further work on a more accurate similarity measure as it has a significant impact on the viability of the indirect parameter selection method.

Increasing the speed of texture synthesis and the associated parameter selection is also an attractive area for future work. A significant speed up would be achieved by implementing Barnes et al.'s patchmatch algorithm [11], with a minor quality loss. Another path of optimization would be to parallelize the complete algorithm and implement it on a GPU. Similar to the work done by Lefebvre and Hoppe [77], who extended Wei and Levoy's 2D synthesis approach [126]. The most computational heavy components of texture optimization are ideally suited to be parallelized and implemented on a GPU. Traversing a high dimensional search space has already been shown to perform much faster on a GPU by Garcia et al. [38], and since texture optimization updates each pixel from a static set of candidate neighborhoods, it could easily be computed using a GPU. As already stated by Manke and Wünsche [83], implementing histogram matching is currently the biggest issue plaguing a direct implementation of texture optimization on the GPU. Kopf et al. have already stated that the histogram must be up to date during the synthesis process. Only updating the histogram in between iterations causes the method to over/undershoot the intended goal.

A possible solution might be to randomly group pixel updates and update the histogram periodically during texture synthesis, instead of after every pixel has been updated.

Acknowledgments

We'd like to thank Johannes Kopf for the past correspondence regarding the texture optimization algorithm. We would also like to extend our thanks to Takeo Igarashi and the University of Tokyo for their collaboration during the creation of this paper. This research was supported in part by the Danish Meat Research Institute.

Chapter 10

Registration-based Interpolation Real-Time Volume visualization

Lasse Farnung Laursen, Hildur Ólafsdóttir, Jakob Andreas Bærentzen
Michael Sass Hansen, Bjarne Kjær Ersbøll

Abstract

Rendering tomographic data sets is a computationally expensive task, and often accomplished using hardware acceleration. The data sets are usually anisotropic as a result of the process used to acquire them. A vital part of rendering them is the conversion of the discrete signal back into a continuous one, via interpolation. On graphics hardware, this is often achieved via simple linear interpolation.

We present a novel approach for real-time anisotropic volume data interpolation on a graphics processing unit and draw comparisons to standardized interpolation alternatives. Our approach uses a pre-computed set of cross-slice correspondences to compensate for missing data. We perform a qualitative analysis using sparse data sets, investigating both visual quality, as well divergence from the ground truth, testing the limits of the interpolation method.

Our method produces high quality interpolation with a moderate performance impact compared to alternatives. It is ideal for reconstructing sparse data sets, as well as minimizing quality loss while scaling large amounts of data to fit on most mobile graphics cards.

10.1 Introduction

A significant number of scientific fields make use of three dimensional data sets. Visualizing these data sets is often advantageous to help gain a better overview of the data itself. Since the advent of non-invasive imaging technology, such as x-ray computed tomography (CT) [58], the medical field has taken advantage of visualizing medical volumetric data. Recently, this kind of non-invasive imaging has also found its way into the food industry, using CT-scanned images of pig carcasses to accurately assess the lean meat percentage [117].

Powerful graphics processing units (GPUs) have provided the computational power necessary to render these three dimensional data sets in real-time. Todays Modern GPUs are programmable and allow for custom code to be executed, enhancing and improving visualization techniques. However, this customizability comes at a cost and is often slower than the operations hardwired into the rendering pipeline.

A central task in volume visualization is converting the discrete data into a continuous signal using interpolation. In our case, interpolation methods can be generally classified as either scene- or object-based [43]. Scene-based interpolation usually makes use of uniform registration, interpolating between values based on their locations within a given image. Commonly used methods in this category includes linear, cubic and truncated sinc functions. Object-based interpolation takes advantage of information contained within the data set, including structure and interconnectivity, to produce more accurate results. A conceptual illustration of the difference between scene- and object based interpolation is shown in Figure 10.1.

Data acquired via x-ray computed tomography is stored in slices, representing periodic measurements across the subject being scanned. The distance between measurements within these slices are often smaller than the distance between the slices themselves. In medical image analysis, this is typically referred to as in-plane vs through-plane resolution. As a result, the voxels (and by extension the data set) are anisotropic.

This anisotropy makes feature aware interpolation methods much more attractive given their likely to reproduce interpolated values that are feature correct.

The built in functionality in modern GPUs only supports *linear interpolation* and *nearest neighbor interpolation*. Like other scene-based interpolation methods, linear interpolation performs poorly and produces artifacts when applied to anisotropic data. The object-based method we present easily outperforms the scene-based methods with a moderate performance impact. Known as *reg-*

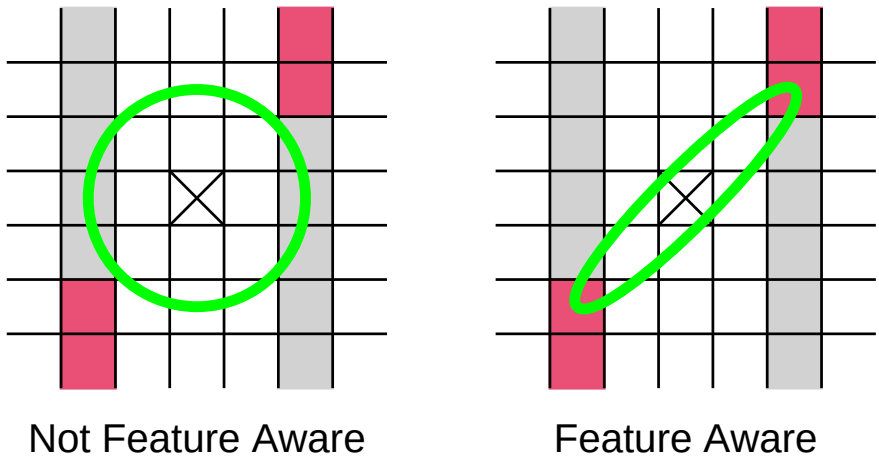

Figure 10.1: A simplified illustration of an interpolating kernel (green) acting on sparse data (grey areas) containing features (red areas). The `x` marks the current value being estimated. Conceptually, the feature aware kernel changes shape to accommodate feature detection, while the non-feature aware kernel retains it shape regardless of surrounding features.

istration based interpolation, the method involves a preprocessing step where the data is analyzed and meta-data is created to aid interpolating the data.

The result is an approach that is ideally suited for modern GPUs, useful for reconstructing sparse data sets as well as compressing large datasets for the purposes of rendering them on GPUs with less memory available.

10.2 Related Work

Interpolation has a long history [84], as its applicability is almost limitless. Grevera et al. give a comprehensive overview of the interpolation methods commonly used on three dimensional image sets, and present a shape-based form of interpolation [43]. Penney et al. present a registration-based method which statistically out preforms both linear and shape-based interpolation [96]. Frakes et al. [36] improve upon the method presented by Penney, both in computational runtime and resulting interpolation. Ólafsdóttir et al. further improve upon the method by registering data in two directions, as well as performing the necessary calculations in a pre-processing step, as opposed to doing them in real-time [69].

Given the flexible programmability of modern GPUs, most scene-based methods for interpolation can be implemented to execute on the hardware. However, the gain in visual quality is rarely worth it, as they incur several costly texture memory look-ups. Even a software implementation of simple trilinear interpolation on the hardware is computationally expensive, compared to the speed of its built-in counterpart.

The issue of accurate interpolation also exists in alternate data representations, such as surface meshes generated from volume data. Moench et al. [87] present a staircase-aware interpolation method to retain surface features while removing staircasing artifacts.

To the best of our knowledge, little prior work exists regarding improving anisotropic volume interpolation, in volume rendering, on GPUs.

10.3 Overview

In Section 10.4, we begin by briefly touching upon the underlying principles behind registration based interpolation for the purposes of arguing for its easy implementation on a modern GPU. Readers unfamiliar with the approach will gain a basic understanding, but we will defer to earlier mentioned papers [36, 43, 69, 96] for a thorough explanation.

Section 10.4.1 details notable implementation details regarding GPU utilization.

The testing protocol along with all the relevant results are detailed in Section 10.5, followed by the conclusion in Section 10.6 which wraps up the work and notes on potential future research.

10.4 Registration Based Interpolation

A trilinear interpolation of the intensity in point $p_{x,y,z}$, located between two 2D slices I_A and I_B, may be formulated in the following manner

$$I_{lin}(p_{x,y,z}) = (1 - \alpha)I_A(p_{x,y}) + \alpha I_B(p_{x,y}), \tag{10.1}$$

where I_A and I_B are the bilinearly interpolated intensity values at the closest

points of the neighboring slices. As will be demonstrated in the paper, this may be suboptimal due to the lack of point-correspondence between these two points. Note, that if the `x` and `y` components of $p_{x,y}$ are not coincident with the voxels in the slices I_A and I_B, bilinear interpolation is applied. α is determined by inter-slice distances (p_z)

$$\alpha = \frac{p_z - I_{B_z}}{I_{A_z} - I_{B_z}}. \tag{10.2}$$

As seen above, trilinear interpolation can be expressed as bilinear interpolation within each slice followed by linear interpolation between slices. A central point of this paper is that we can obtain a much better reconstruction by replacing this linear interpolation between closest points with an interpolation that takes slice correspondences into account. Proper correspondences may be achieved automatically using image registration. Hence, the basis for the registration-based interpolation approach is the set of 2D image registrations between each pair of slices, both ways. This results in a deformation (correspondence) field represented as a set of two dimensional vectors in each point. For the experiments in this paper we have used registration based on B-splines [103] with sum of squared differences as a similarity measure. Note however, that given that the resulting deformation fields are sufficiently smooth, the proposed interpolation scheme is independent of the choice of registration algorithm.

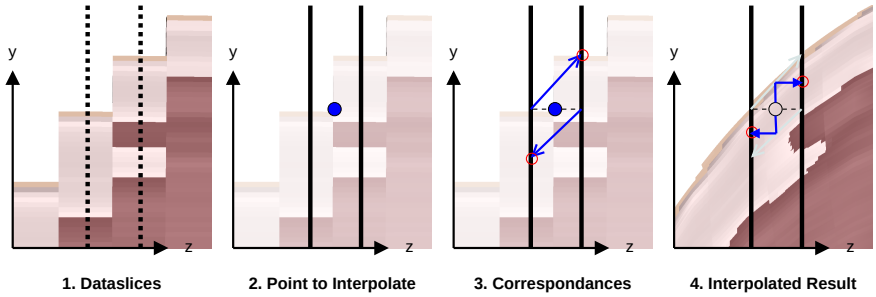

Figure 10.2: A simplified illustration of the registration interpolation method. 1. A cross section of rendered volume data. The dotted lines represent the actual location of two data slices. The data is initially shown using the nearest value filter (no interpolation). 2. The point we wish to interpolate is shown. 3. The deformation fields reveal the correspondence vectors for the immediately adjacent voxels. 4. Since our point to interpolate is exactly half way between our data slices, we only use one half of each correspondence vector to find the appropriate voxels to finally interpolate in between linearly.

Figure 10.2 shows the interpolation process broken down into four steps. The

data point we wish to acquire has been placed directly in between two known slices of data, for the purposes of this example.

More generally, given a point $p_{x,y,z}$, the registration based interpolated data is calculated by

$$\begin{aligned} I_{reg}(p_{x,y,z}) = (1-\alpha)I_A(p_{x,y} + \alpha\varphi_{BA}(p_{x,y})) \\ +\alpha I_B(p_{x,y} + (1-\alpha)\varphi_{AB}(p_{x,y})). \end{aligned} \tag{10.3}$$

where I_A and I_B are the slices closest to the point we're interpolating [69]. The deformation fields from I_A to I_B and I_B to I_A are represented by φ_{AB} and φ_{BA}, respectively. Note that as opposed to Eqn. 10.1, not only intensity but the neighbor point position, given by the amount of deformation, is determined by α.

Figure 10.3 illustrates the aforementioned variables, apart from the deformation fields φ_{AB} and φ_{BA}, whose contribution is visualized by the green arrows.

10.4.1 GPU Implementation Notes

Choosing the right data type for representing the volume data is important, in order to make the most of the memory afforded on the GPU. The data was generously provided by the Danish Meat Research Institute (DMRI)[28] in the form of 16-bit integer values in the industry standard dicom [91] file type. Adhering the latest OpenGL standard [107], the ideal storage format is the recently introduced 16-bit floating point values format, which supports non-clamped values and does not require rescaling of the original density measurements. The original density values from the pig carcass span from `-3024` to `2447`, hence there is minimal loss of precision by using the half-precision floating-point format.

In addition to the volume data, the associated deformation fields, described in Section 10.4 must also be stored locally on the GPU for optimal performance. It turns out that they are an ideal fit for an already standardized texture format. Since every interpolated value needs two displacement vectors (as visualized in Figure 10.2 and 10.3), a texture containing the entire color spectrum as well as the alpha channel (`RGBA`) is ideal for its representation. By storing the `x` and `y` components of the two vectors in the `R`, `G`, `B`, and `A` components respectively, only a a single texture look-up to retrieve both displacement vectors, as well as two successive look-up to determine the actual density values to interpolate in-between.

The deformation fields are traditionally represented via 32-bit floating point

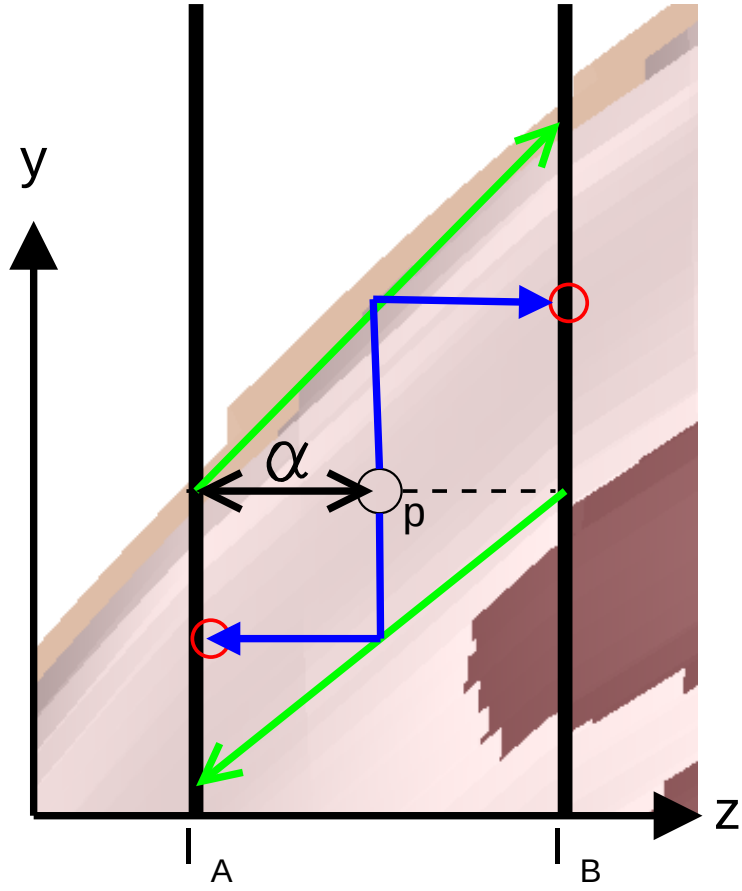

Figure 10.3: A magnified annotated version of fig. 10.2, sub-image 4.

values. This ensures a precise representation at the cost of high memory usage. Most medical datasets maintain a certain level of spatial locality, it is therefore natural to assume that the displacement vectors have a limited range within which they point. We will therefore analyze the impact of having the vectors scaled to be represented via less precise 8-bit signed integer values.

10.5 Testing

In this section we begin by outlining our testing setup and concrete testing protocol. The obtained results are presented along an evaluation of the qualitative impact.

10.5.1 Setup and protocol

Our testing was conducted primarily on an Alienware m17x laptop, utilizing two SLi connected GeForce GTX 280M graphics cards, each having 1 gigabyte of memory available. We used two three dimensional image sets of two different pig carcasses, provided by the Danish Meat Research Institute (DMRI)[28] as data, for our testing purposes. The first dataset was specifically scanned to be isotropic and serves as ground truth for our quantitative testing, with a size of `212x512x1662` in the `x`, `y`, and `z` dimensions respectively. The other dataset is representative of a typical sparse scan performed in an actual slaughter house, with a size of `256x480x134`. The displacement fields are calculated in `Matlab` using code kindly provided by Ólafsdóttir et al. The real-time rendering is preformed using `OpenGL` in conjunction with `C++`.

Our comparative evaluations are limited to the registration based interpolation method and the standardized trilinear interpolation. It is tempting to consider including a wider array of standardized interpolation methods, such as tricubic interpolation. While this type of interpolation is likely to provide more pleasing visuals when compared to trilinear interpolation, it is still a scene-based interpolation method and as such will render results without the accuracy that structural information can provide. In addition, tricubic interpolation and other methods utilizing a wider array of samples are technically unfeasible since they require many computationally heavy texture look up calls. Section 10.5.5 describes the performance impact, proving this, and similar methods, to currently be unusable in a real-time context.

We use the isotropically scanned dataset as ground truth and create 15 alternate sparse datasets based on it. Each new dataset has an increasing number of slices removed following the pattern visualized in figure 10.4.

Removing slices in this pattern ensures that there are as few periodically reoccurring slices as possible, while still maintaining the same number of missing slices between retained slices. This minimizes potential bias by maximizing the variation of data contained in the slices that are to be interpolated. An exception being the very first slice which is always retained. Figure 10.5 is a

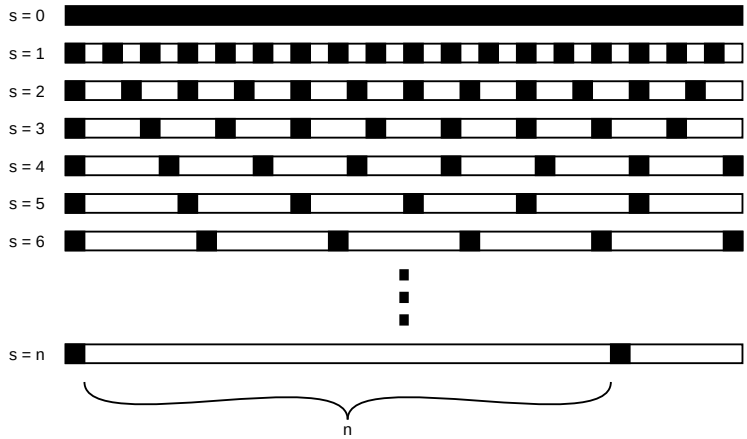

Figure 10.4: Visualized slice reduction. The black portions represents retained data. As n (sparsity level) increases, the data becomes more sparse.

comparative chart of the size difference between the original dataset and the more sparse data sets, along with displacement data. Note that we mainly disregard the least sparse data set (sparsity level 2) during testing, as it essentially just replaces the removed slices with even more voluminous displacement data, nullifying any benefit gained by using registration based interpolation.

Since volumetric data sets derived from pig carcasses contain a large amount of air, we apply a mask to filter out voxels which are uninteresting during quantitative testing. We threshold the entire volume and retain voxels which are sufficiently dense to be identified as pig matter. However, to avoid risking missing transitional voxels from air to matter, we perform a simple mathematical morphological dilation operation to include all voxels adjacent to an already included voxel.

Because we have the original isotropically sampled volume, it is possible to compute the difference between an interpolated value and the original true voxel value (from a removed slice) at the precise location of the given voxel. This allows us to gage the quality of our interpolation method as well as compare it to other interpolation methods.

We are also interested in knowing how well the registration method [69] used to compute the deformation field, performs. In other words, validating the method by performing a comparison in between deformation fields from differently sparse data sets. Ideally, we would obtain the same result if we first track a feature from slice 1 to slice N, one slice at a time, and then compare the sum of these

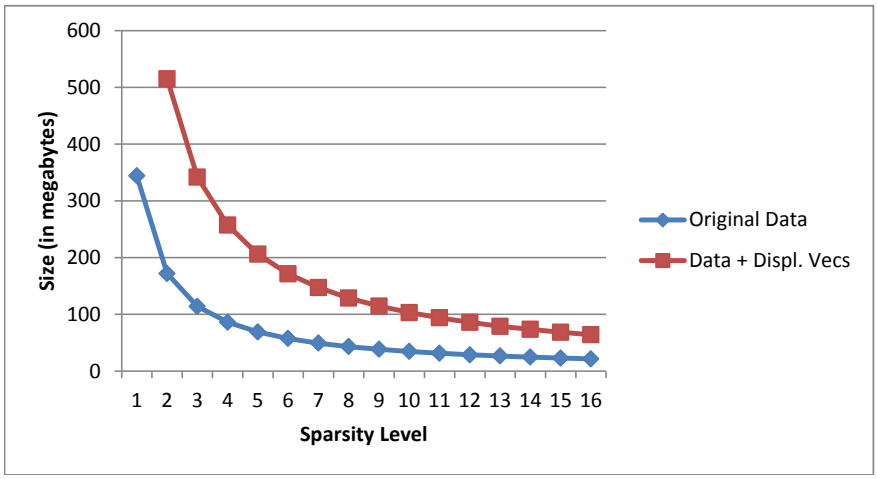

Figure 10.5: Comparative chart of the sizes of the originally isotropic pig carcass data set and the sparse versions, along with potential deformation field data.

deformation vectors to the one produced by the direct registration of slice 1 to slice N. This hypothesis is easy to verify by comparing accumulated deformation fields from a less sparse data set, to the deformation field of a more sparse data set. The concept of tracking just a single feature is visualized in Figure 10.6.

Having determined the deviation of each interpolation method from the established ground truth, we subjectively evaluate the visual quality of the method applied to our pig carcass data sets.

10.5.2 Quantitative Evaluation

We begin our analysis by comparing both the hardware supported trilinear interpolation method, and the registration based method, to the ground truth provided by our isotropic data set. Figure 10.8 shows the effects of interpolation values from an increasingly sparse data set.

As the results in Figure 10.8 depict, the registration based method is closest to the ground truth on every tested level of sparsity.

The results of the 8-bit integer registration based method shows that a reduction in precision of a factor of four, has little quantitative impact on the performance of the registration based interpolation.

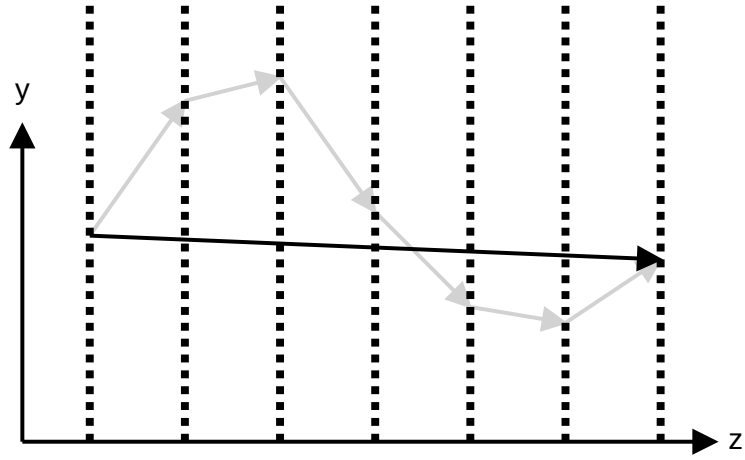

Figure 10.6: An illustration of an ideal scenario of displacement vectors from two differently sparse sets. The light gray arrows are derived from a less sparse set, and in this case add up to the displacement vector calculated from the more sparse data set. In this ideal scenario, the summation of less sparse displacement vectors always conform to the displacement vector from the very sparse data set. In a less ideal scenario the gray arrows would not sum to the same location as the black arrow.

10.5.3 Multiple Deformation Slice Correspondence Comparison

To further validate the registration based interpolation method, we compared deformation vectors from our datasets with a sparsity level of 3 and 15. Using the approach described in Section 10.5.1. Each deformation vector from the sparse data set is compared with the sum from the corresponding displacement vectors, visualized in Figure 10.6, in the less sparse data set. Subtracting the vectors from each other yields the difference in offset, and the length of this vector reveals how accurate the registration based interpolation is at determining identical correspondence between these two differently sparse data sets. The severity of the offset is determined by the real-world sample distance in between voxels, which in our case is approximately 1 `mm` in the original isotropic dataset.

In one direction, the mean difference in length was just a little over a single voxel with a value of `1.11015`, where as the other direction had a mean length difference of `3.14534`. This means that there is an approximate average deviation of 1 millimeters and 3 millimeters respectively.

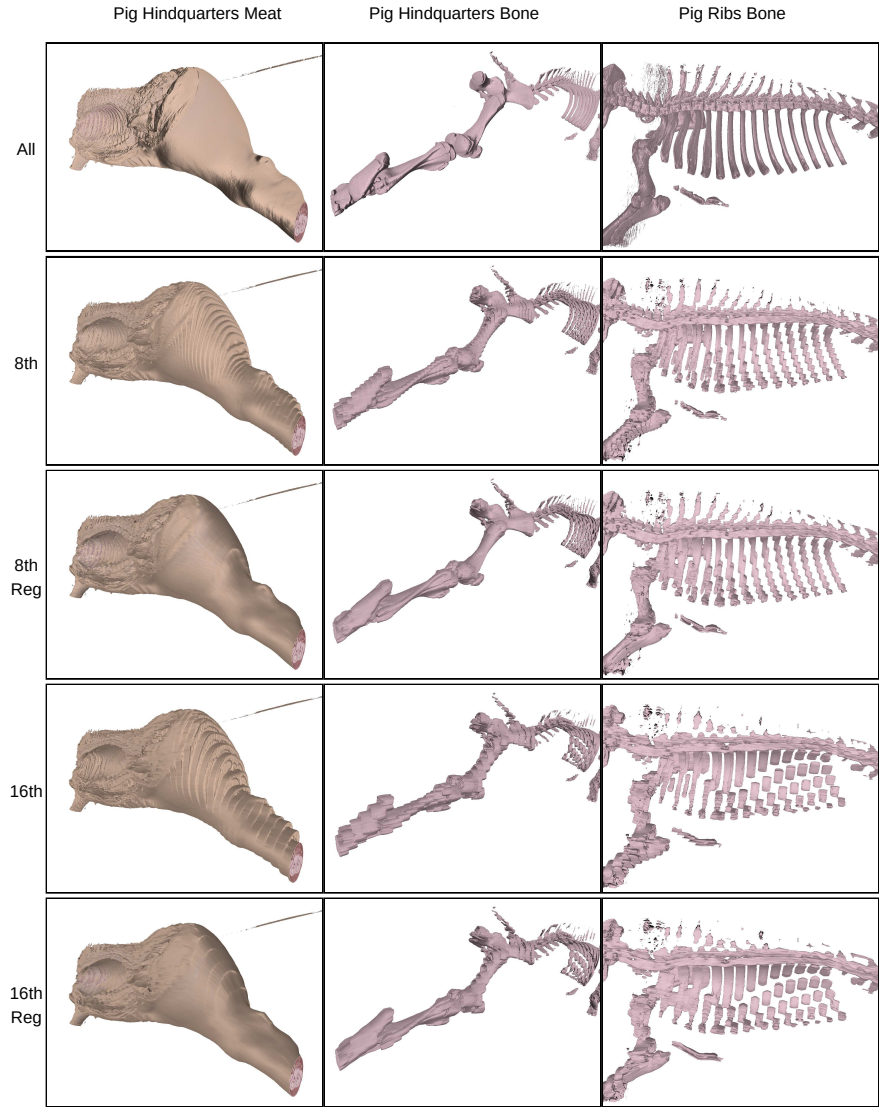

Figure 10.7: Five sets of real-time volume rendered images of the originally isotropic pig dataset, rendered from three different perspectives. The first set is the densest volume data set, followed by a sparse set containing every 8th slice of the original using trilinear interpolation, followed by the same data set using registration based interpolation. The final two sets contain every 16th slice of the original data set using trilinear and registration based interpolation.

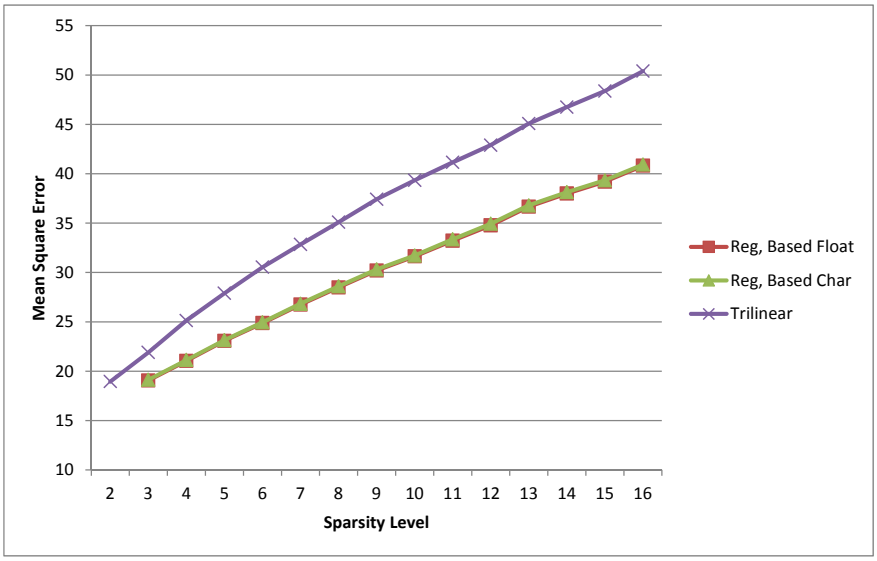

Figure 10.8: The Measured Mean Squared Error measured by by comparing the interpolated density values to the ground truth.

10.5.4 Qualitative Evaluation

The 32-bit registration based interpolated visualization is almost visually indistinguishable from its 8-bit counter part and has therefore been omitted from the Figures presented in this Section.

Figure 10.7 shows a comparative visualization of the originally isotropic data set along with 2 sparser versions, each containing three different perspectives. The registration based interpolation method visually surpasses the trilinear interpolation method by far. In most cases, the displacement field succeed at masking staircasing artifacts caused by sparse data.

One of the most visually striking differences between all the 15 sparse datasets, and the complete one, is that a lot of fidelity is lost due to poor surface normals (calculated using the central differences method). Even when every 2nd slice is retained. The images make it clear that features spanning across several slices are much better recognized by the registration compared to features that are almost parallel to the slice plane. The ribs shown in the right column of Figure 10.7 are especially susceptible to incorrect interpolation once the data set becomes sparse enough. Ribs start melding together at a sparsity level of 7 (every 7th slice retained).

10.5.5 Performance Benchmark

We perform a simple performance benchmark by measuring the average rendered frames per second over a 60 second time span. To ensure a representative measure, the volume is constantly rotating to avoid any bias introduced by only rendering a slender side of the pig carcass data set. In addition, each interpolation type is performed using unique shader code, due to the high performance impact of introducing just a single branching statement in an often used function, such as intensity look-up.

Using hardware supported trilinear interpolation each color value calculated along a traced ray is comprised of 7 texture look ups. One for the local intensity, and an additional six for to calculate the normal via the central differences method. When using registration based interpolation, the amount of texture look ups is tripled. For each built-in trilinear interpolation look-up, the registration based method performs one to receive the deformation field, and two additional look ups to obtain the to-be-interpolated intensity values. In the case of tricubic interpolation the number of texture look ups, when compared to hardware support trilinear interpolation, is increased by 64 times. This total of 448 texture look ups drops the average frames rendered per second drops far below 1. Our empirical testing lead to a single frame being rendered after approximately 9 seconds. Consequently, the tricubic interpolation and similar sample heavy interpolation approaches are computationally unviable in real-time.

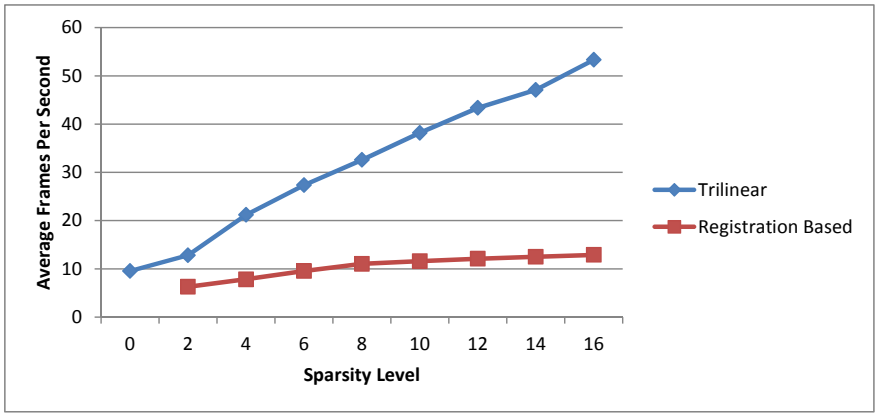

Figure 10.9: Average frames per second, measured over a 60 second time period, for both hardware supported trilinear interpolation, and registration based interpolation.

As the graph shows in Figure 10.9, the built-in trilinear interpolation is a clear winner in any isolated case. This is not especially surprising given the 1-to-3 ratio of difference in texture look ups, along with some additional operations.

More interesting is the point at which the registration based method outperforms the dense data set, which happens when approximately every 8th slice is retained.

10.6 Conclusion

The registration based method presented in this paper has been proven to be a viable real-time rendering interpolation alternative compared to the built-in trilinear interpolation. While it cannot compete with the speed of the built-in trilinear interpolation, it offers real-time performance as well as a significant reduction in memory use, while maintaining a higher quality visualization. The registration based method is, per definition, likely to outperform any method which is not feature aware.

We have determined that there is a significant correspondence between features when in our case when comparing two sparse datasets derived from the same isotropic dataset.

Using this approach allows for larger volumetric data sets to be visualized on GPUs with less memory available.

10.6.1 Future Work

It would seem likely that certain data slices are more essential than others to maintaining acceptable results using registration based interpolation. The difference in deformation accuracy based on direction noted in Section 10.5.3 corroborates this hypothesis. Given an isotropic data set acting as ground truth, it would be interesting to develop an optimizer which removed and re-added slices in order to determine the optimal set of retained slices while minimizing memory usage. Essentially creating an optimized volume compression format.

We've determined that our dataset yields deformation fields that correspond to the same features with minor deviation. It would be interesting to apply the multiple deformation slice correspondence comparison to other datasets and determine which features yield worse or better correspondence fields.

Although sample heavy interpolation schemes are computationally too demanding to execute in real-time on modern GPUs, it would be interesting to evaluate the performance/quality trade-off involved in improving the in-slice interpola-

tion method. The registration based method relies on standard bilinear interpolation, assuming a sufficiently dense data set, but perhaps bicubic interpolation would yield visually improved results with an acceptable performance hit.

Acknowledgments

We extend our thanks to the Danish Meat Research Institute (DMRI) [28] for their support as well as for providing us with the data sets of the scanned pig carcasses.

Chapter 11

Pig Product Prototyper: Cutting interface design

Lasse Farnung Laursen, Jakob Andreas Bærentzen, Takeo Igarashi
Michael Kai Petersen, Line Katrine Harder Clemmensen
Bjarne Kjær Ersbøll

Abstract

With the help of industry experts we developed Pig Product Prototyper, an application intended to aid in the communication process between producer and retailer when developing new meat products for a constantly evolving market. The application interface allows the user to make planar cuts to a virtual pig formed from CT-scans of a real-world pig carcass. We perform a comparative study of two different controller interfaces for the application, one being a traditional mouse and keyboard input, and the other a six degrees of freedom haptic feedback device. The goal was to assess usability issues and overall usability for the target group concerning both interfaces.

The accurate depiction of pig anatomy can guide trained professionals to re-create standardized pig products. The results of the usability test with sales personnel do not lean significantly in favor of either interface, despite the participants expressing favor towards one interface. This stalemate carries a significance in regards to the more alien interface introduced to the users. We report on the development process and observed user experience regarding the two interfaces.

11.1 Introduction

Approximately 90% of the danish meat industrys profit is derived from exporting products. Meat industries from competing nations, such as Germany and Spain, are catching up to Denmark in terms of product quality and price. Thus, the danish meat industry is under continuous pressure to find innovative ways to improve service and production, thereby minimizing costs and maximizing efficiency. One such area of improvement is the existing product development cycle between customer and meat producer.

A number of meat products are created using internationally recognized standards that many meat producing companies apply. However, in a number of situations these standardized products do not meet the needs of the buyer and must be further refined, or even created from scratch. In this case, the meat producer and meat retailer communicate in person and via long distance in order to settle on terms agreeable for both parties involved. Visual aids (static images) are occasionally used, and at some point during the entire process technical staff, trained in the slaughtering of carcasses, provide expert knowledge. It can take several meetings before both parties agree to the final product, and in the course of these meetings, real prototypes are produced from carcasses that can no longer be reused. The process, as a whole, is iterative, unstandardized, and destructive. Due to the lack of established standards, there is no fixed time frame for this development cycle, but on average it takes anywhere from a few weeks to a month until the final specifications are in place.

We communicated with experts within the danish meat industry to assess their needs regarding virtual product development, and designed the Pig Product Prototyper (PPP) as the first prototype milestone. We present the design process, and evaluation of the Pig Product Prototyper developed in collaboration with the Danish Meat Research Institute (DMRI) [28] and Danish Crown. Using the application in the communication process allows for a less wasteful and more efficient product design cycle, as well as supporting long distance communication without the introduction of additional ambiguity. The prototype was tested with the intended target group, consisting of the sales staff at Danish Crown.

The PPP renders a virtual pig carcass, based on CT-scanned data from a real pig, and allows the user to create, modify and delete planar cuts. A screenshot from the application is shown in Fig. 11.1, showing the CT-scanned pig data cut using a single plane. Planes are created, moved and rotated in real time to fit the users needs. We designed two separate interfaces: one using a traditional mouse input with the keyboard, while the other uses a six degrees of freedom haptic feedback device (Phantom Omni haptic feedback device © Copyright Sensable

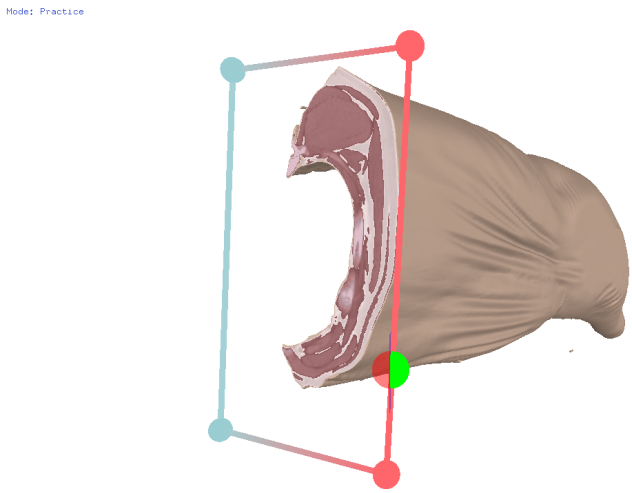

Figure 11.1: Pig Product Prototyper. Showing a direct volume rendered pig carcass with a single cut plane, partially selected via the cursor.

Technologies, Inc. [94]). In the case of both interfaces we prioritized simplicity over functionality in an effort to allow for a quick and simple introduction during testing.

Since none of the interactions provided by the PPP specifically call for haptic interaction, one might ask whether a unfamiliar device such as the Phantom Omni is justified. However, we hypothesize that the ease with which spatial positions and orientations can be input via the Phantom should level out the learning curve for novice users. Thus, the Phantom might indeed be preferred by (some) users and this is a part of what we try to investigate in the presented paper.

11.2 Related Work

This work contributes primarily to the areas of volumetric modelling interfaces and usability testing. We survey related research on these topics and note similar projects.

A significant portion of the decisions and actions taking during the evaluation

of our interface, is derived from previous research [101, 115]. There has been a number of previous works regarding the evaluation of multi-dimensional input devices, specifically the Phantom Omni.

The steps taken during an evaluation of a given interface is highly dependent on the type of interface and the intended goals. It is unlikely that another project matches the exact pre-requisites and goals of this (or any other) project. Harders et al. [49] perform a study in between multiple 6DOF haptic interfaces, which includes the Phantom Omni, and determines none of them to be significantly superior to any of the others. Plimmer [99] compares a number of input devices, including both the mouse and Phantom Omni, when used in pen-dominant software tools. She comments on existing usability issues with pen-based forms of interactions, but also notes that users tolerate the inefficiency of the pen in exchange for convenience. Yhu and Lee [134] present the technical aspects of a virtual prototyping haptic interface using a 5DOF haptic input device.

A large body of research exists regarding volumetric modelling. Galyean and Hughes [37] present a volumetric sculpting interface using a variety of tools, allowing the user to create a rough replica of an object. A more precise volumetric sculpting tool is presented by Wang and Kaufman [121]. The FreeForm® systems [106], created by Sensable®, is noted by Gregory et al. as probably being the first 3D digital modeling tool to allow users to express themselves creatively using 3D-Touch™. Gregory et al. present their own Haptic-enabled 3D interface, using a Phantom Omni, called inTouch [42].

To our knowledge, we were unable to locate a previous comparison between both the mouse and a 6DOF input device.

11.3 Design Process

The idea of a meat product prototyping tool originated from the danish pig meat industry to serve as an additional competitive advantage. To establish the needs of the PPP application, we interviewed employees associated with the development of new products to gain insight into the unstandardized process. We identified the main areas of requirements as follows:

- **Product Reality** - A key factor for the success of the PPP is the realism with which it can depict the anatomy of the pig. The anatomy itself can almost be described as a cutting blueprint, which experts use as a guide to properly produce standardized products, despite biological variations in between pigs. A dynamic 3D pig carcass is only useful, if the same

anatomical aspects of the pig can be recognized by experts, as on a static photo.

- **Factory Reminiscent Cuts** - The procedure of slaughtering a pig and producing consumer products involves a number of steps. In general, the first half of the procedure is primarily automated using machines, apart from the actual killing, which has to be done by a certified butcher. The second half is usually less automated and more done by hand, depending on how close to a consumer product, the final result is. The cuts committed by the PPP should correspond to real world applicable cuts, committed during the process of actual product development.

- **Ease of use** - Although this is practically a preferred requirement for any application, the relative ease of use is quite dependent on the intended target group, and their technical prowess. The target group in this case consists primarily of the sales staff which has received no formal training related specifically to the prototyping of meat products, and span a wide age gap from the twenties to late fifties (24-58).

The advancement of CT-scanning integration into modern abattoirs, made it ideal to leverage volume data for the purposes of realistic product reproduction. Even a fairly low resolution scan of a pig carcass still encompasses all the significant anatomy. Using direct volume visualization as our approach to realistic product visualization, we designed a few prototypes of various interaction techniques. Since a number of the most common products are created using planar cuts, we decided to limit the design to only allow the user these simple planar cuts. Early versions allowed the user to mimic the movement of cutting through meat by drawing a line along the pig carcass surface and estimating the best fitting plane. We also prototyped versions allowing the user to feel the resistance of the different types of pig tissue contained in the carcass: fat, muscle, and bone. Our empirical testing lead us to believe that this complexity only served to complicate what should ideally be as simple an interface as possible: One allowing the user to quickly and easily create product prototypes.

Having the prototype interface completed, we are interested in evaluating how well the prototype performs in terms of both usability, as well as the aforementioned requirements. As previously mentioned, the main target group for this application is the sales personnel involved in communicating with clients regarding the development of product prototypes. However, this group of people do not possess the expertise with which to evaluate and create virtual products matching the real counterparts. This knowledge must be provided by an educated butcher. The sales personnel are, however, the ideal test participants for usability testing, as they are the end users of the PPP. Tasking these participants

with the recreation of well known products simulates a realistic application use which allows us to gage usability issues and measure performance.

Before submitting the prototype design to a usability study, we subjected it to a pilot test to determine the most glaring issues with the first iteration. A number of significant issues were observed, both with the design as well as the testing procedure to warrant a second pilot test. The second pilot test revealed minor issues which were addressed, at which point the prototype was ready for the actual formative usability study.

11.4 Implementation details

As previously stated, the use of volumetric data is essential in the virtual recreation of the pigs anatomy. The PPP is essentially a much more versatile substitution for the static pictures occasionally used during a product negotiation. To maintain high responsiveness from the interface, the visualization of the volume data relies heavily on a set of modern GPUs (two GeForce 280m in SLi). This computational power generally provides an interactive frame rate at approximately 25 frames per second (FPS). However, if the pig carcass fills up the entire view window, the application is still subject to unwanted slowdowns.

We implemented two features to keep the frame rate and by extension the responsiveness of the application as high as possible:

- **Low-High Quality Mode Rendering** - When either the pig carcass or any of the planar cuts is moved, the rendering engine switches to a lower quality rendering mode in order to maintain a high frame-rate. Once the user ceases to interact with either the pig carcass or any of the planar cuts, a high quality version of the current perspective is rendered. The quality difference is visualized in Fig. 11.2.
- **Buffered Volume Rerendering** - Every frame, the volumetric rendered pig carcass is rendered into a buffer. This ensures that the entire scene only needs to be redrawn if the pig or any associated cut planes are altered.

  As long as neither the pig carcass nor the cutting planes are moved or rotated, the user interface is combined with the pre-rendered image of the pig using the depth buffer from the volume rendering.

It should be noted that the interpolation algorithm used in conjunction with direct volume rendering gives an appearance of skin on any uncut surface. How-

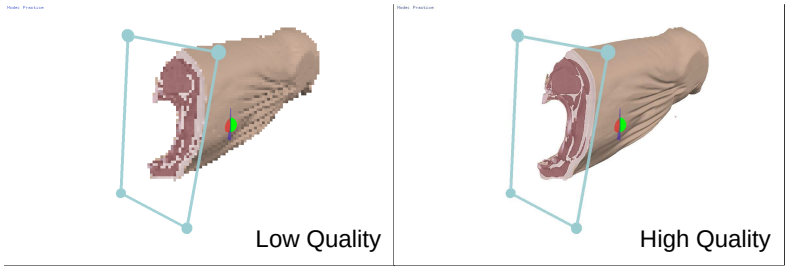

Figure 11.2: Two screenshots taken from the same camera angle in high- and low-quality mode respectively.

ever, the shape of the surface and it's notable features still correspond to that of a real pig carcass.

Despite the fact that modern GPUs are highly programmable compared to their older counterparts, there are still a number of restrictions in place when executing a large portion of the render code directly on the GPU. One such restriction is the limitation on the complexity of the variable containers used on the GPU cores. Designing the application to utilize planes as its primary cutting geometry allowed us to leverage the GPUs design to our advantage. In its most compact form, a plane can be represented by a normal vector and a scalar value. This compact "four value" representation allowed us to use simple predefined structures in place on the GPU cores.

We also implemented a simple binary mask to allow the user to create completely custom cuts into the pig volume, but this functionality was determined to be too complex for the simple interface we wanted to compare in our usability study.

11.5 Interfaces

Visually, the two interfaces are almost identical. The user is presented with a direct rendered volume consisting of CT-scanned data from a real-world pig carcass. The user has the ability to create, modify, and delete cut planes. Each cut plane consists of a frame and corners surrounding the pig volume, as illustrated in Fig. 11.1.

Apart from the visual representation of the pig carcass depicted in Fig. 11.1, the user also has the option to render an x-ray-like perspective of the volume data as visualized in Fig. 11.3. For both interfaces, the user has to press the `space bar` on the keyboard in order to switch between them.

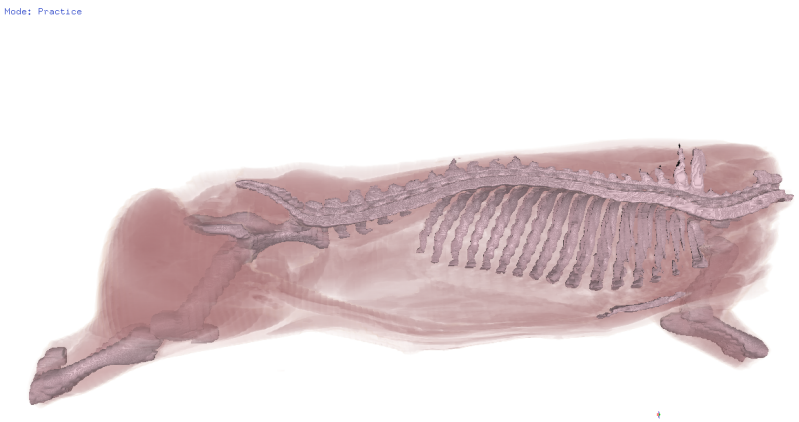

Figure 11.3: Pig Product Prototyper visualizing the volume using an x-ray-like renderer.

11.5.1 Mouse Interface

The mouse interface allows the user to manipulate the position and orientation of the pig carcass exclusively using the mouse. Any movement or rotation requires the simultaneous pressing of a mouse button and moving of the mouse. The left mouse button allows the user to rotate the pig carcass in fashion similar to the virtual sphere interface described by Chen et al. [22]. The right mouse button allows the position of the pig to be translated along the current vertical and horizontal viewing axis. The middle mouse button allows the user to move the pig carcass closer or further away.

To create a cutting plane, the user has to place the cursor on the intended part to by cut and press the `enter` key on the keyboard. This creates a cut plane with the same orientation as the small blue plane visualized as part of the cursor. The orientation and position of the plane can then be changed via controls similar to moving the pig. The user is free to select either a corner or an edge of a cutting plane using the cursor. By then clicking the left mouse button coupled with mouse movement the user can rotate the plane. The right mouse button will instead allow the user to move the plane along the orientation of the pig carcass. Visualizations of the various interactions are shown in Fig. 11.4.

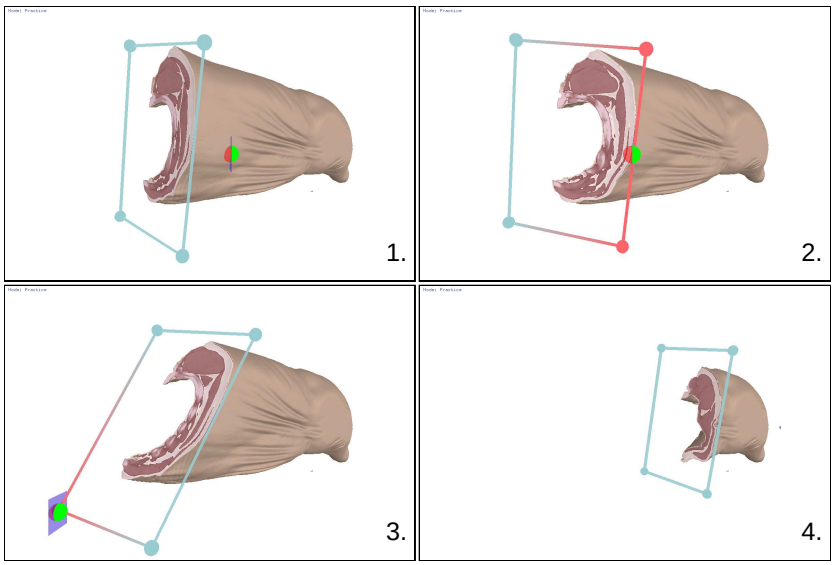

Figure 11.4: Four different screenshots depicting mouse interaction with the PPP application. 1. A single cut plane applied to the pig carcass. 2. Left mouse button interaction with an edge of the cut plane allows the user to move the two edge endpoints around the center of the cut, effectively shifting all four visualized corners. 3. Left mouse button interaction with a corner allows the plane to be rotated along the axis running between the two neighboring points. 4. Right mouse button interaction with any part of the plane allows it to be translated along the pigs current orientation.

Due to the limited number of degrees of freedom presented by the mouse, the freedom of rotation and translation of the cutting planes was designed to be restricted around the pig volume. Since the cutting planes remaining centered around the pig, the rotation and translation of the cutting planes is slightly different than the rotation and translation of the pig carcass. The cutting planes move and rotate in relation to the current perspective the user have of the cutting plane.

Deleting a cutting plane is accomplished by translating the plane far enough along its normal so that it's no longer intersecting any of the visible pig volume.

11.5.2 Phantom Omni

The Phantom Omni based interface provides the user with direct interaction of the pig carcass and associated cutting planes. The device has two buttons on

the front end of the pen, as shown in Fig. 11.5. The button closest to the tip will be referred to as the forward button, and the other as the back button.

Figure 11.5: Photo of the Phantom Omni haptic feedback device from Sensable, used with permission by Sensable Technologies Inc.

The front and back button manipulate cutting planes and the pig respectively. By holding down the back button, the orientation and position of the pig will change according to how the Phantom Omni pen is rotated and moved, as long as the button is held.

The forward button is used to introduce and manipulate existing cutting planes. If the cursor is not near a plane when pressed, a new cutting plane is introduced with the same orientation as the small blue plane on the cursor. If the cursor is near an existing plane and the button is pressed, its rotation and translation will mimic that of the Phantom Omni, as long as the button is held.

This one to one relation of movement and orientation between the Phantom Omni and the controlled elements (e.g. pig carcass, planes) follows the principle of least astonishment [104]. The users would expect the elements to move and rotate in the same fashion as they move and rotate the pen on the Phantom Omni.

Deleting a plane with the Phantom Omni is done in a similar fashion to the mouse, by dragging the plane along its normal until the cutting plane no longer intersects with the visualized pig volume.

The Phantom Omni also provides haptic feedback when the cursor collides with the pig carcass, effectively allowing the user to rest the cursor on the surface of the pig. The cutting planes have a light magnetic effect to them, attracting the cursor if it is close by.

11.6 Evaluation

11.6.1 Expert Product Creation

We gathered expert knowledge from a trained butcher for the purposes of recreating and collecting cutting data of five standardized products. Since the primary goals were to collect expert knowledge and assess product reproduction, an expert user cooperated with the trained professional butcher in recreating the standardized products. The expert user interacted with the PPP interface guided by the professional who provided instructions as to where to apply the cuts to the pig carcass. Photographs of the five products along side the virtual reproductions can be seen in Fig. 11.6.

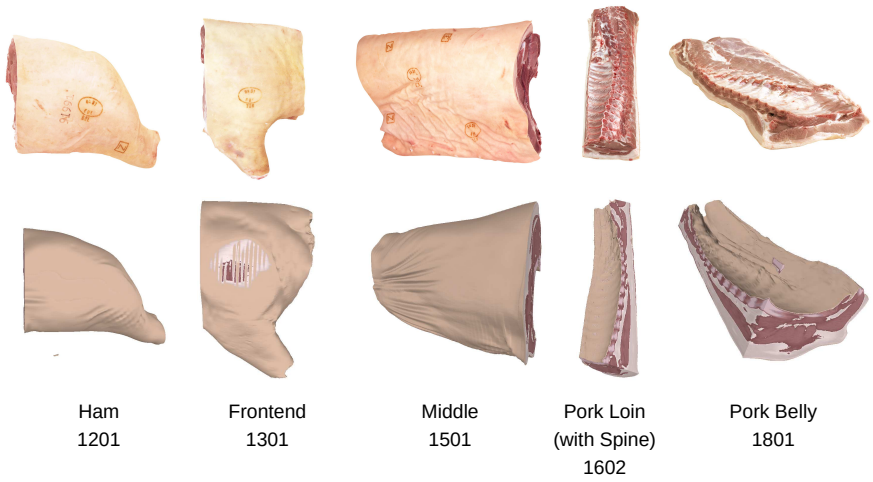

Figure 11.6: The five products shown alongside their virtual counter parts. The virtual products shown in the bottom are created using the expert cuts. The most striking visual differences are caused by a lack of physical simulation, most notable in product 1801, where the cut flesh does not even out at the end like in the real world. The other notable difference is skin appearance caused by interpolation most notable in product 1602 and 1801.

The volumetric representation of the pig carcass easily allowed for the trained professional to recognize both bones and muscle groups, and made significant use of both the standard and x-ray-like rendering of the pig carcass. The low quality rendering mode did not inhibit the butcher from recognizing significant anatomy used as a guide to apply the cuts, which allowed the cuts to be placed quickly and precisely aided by the expert user. The previously mentioned visual artifact of malplaced skin color also did not negatively affect performance.

It is possible to provide a rough general measurement of the accuracy of the applied expert cuts, by examining the first three products visualized in 11.6. These three products together make up the entirety of the pig carcass from which they are each individually cut. Out of the `4261045` voxels comprising the pig carcass, the first three expertly cut products together make up `4262697`, yielding a difference of `1652` voxels, less than `0,1` %. It is, of course, possible that the first product includes the whole pig, and the other two cut it away completely, yielding the best accuraccy, but as Fig. 11.6 shows, this is not the case.

11.6.2 Formative Usability Study

In collaboration with DMRI [28] and Danish Crown a formative usability study was conducted to evaluate potential usability issues as well as ease-of-use of the mouse and Phantom Omni based interfaces. A total of 8 participants, 7 male and 1 female, volunteered from the sales staff at Danish Crown to participate in the test. General information for each participant is listed in table 11.1. Each of the participants use computers in their day to day tasks and none of them had had previous interactions with either of the interfaces. Nor had any of the test participants used a Phantom Omni or similar haptic feedback device prior to testing. The study took place at the Danish Crown headquarters in Randers in a meeting room depicted in Fig. 11.7.

As Tullis and William note [115] (p. 54), performing a comparative study of alternate designs carries a few notable drawbacks with it. There is a high likelihood of learning effect in between designs, and performing the same task with multiple designs can quickly become tedious.

We believe the detrimental effects of the comparative study using the same volunteers is minimal in our case. Although the visual appearance of the interface is nearly identical, the interaction style for each of the interfaces is radically different given the difference in input controllers. The tasks we required the participants to perform were all of a simple nature and did not involve ordered logical solving in order to succeed. Thus, repeating them with a different con-

Participant	Age	Gender	First Use	Preference
1	32	Male	Mouse	P. Omni
2	49	Male	Mouse	P. Omni
3	53	Male	P. Omni	Mouse
4	28	Male	P. Omni	Mouse
5	45	Female	Mouse	Mouse
6	37	Male	Mouse	P. Omni
7	24	Male	P. Omni	P. Omni
8	58	Male	P. Omni	P. Omni

Table 11.1: The total number of participants in the usability study, along with their age, gender, which interface they were presented with first, and which they expressed a preference for.

troller posed a new challenge. Finally, the total number of tasks was small so as to minimize tedium.

Despite these facts, we still performed counterbalancing and had four participants begin the test using the mouse interface, and the remaining four begin the test with the Phantom Omni interface. The tasks themselves were not counter balanced, as they had a logical order in which they required completing.

For each of the two interfaces, the participant partook in the following phases:

1. **Introduction** - The user is presented with an instructional video demonstrating the use of the interface, followed by a short interactive session where the user can perform all of the possible actions in a practice mode.
2. **Navigation Practice** - The user is presented with a short series of challenges, described in the sub-section below, to aid and assess the users approximate skill level with the interface.
3. **Product Creation** - The user is instructed to create five standardized pig products, for the purposes of simulating a realistic use case. This phase of the usability test is described in a later sub-section.
4. **SUS Questionnaire** - The user is given the System Usability Scale (SUS) Questionnaire [16] to fill out.

During the testing sessions, the participants are free to ask questions as well as encouraged to voice any thoughts regarding either interface, both positive and negative. Following the completion of all the tasks with both interfaces, the participant is asked to choose a preferred interface as well as the reasons for said choice.

Figure 11.7: The meeting room housing the test set up for our usability test, complete with complimentary refreshments to motivate participants.

11.6.2.1 Navigation Practice

The user is shown one of five navigational frames which has to be aligned to fit the corners of the screen, as visualized in Fig. 11.8. The green corners have to be aligned to the bottom and the blue corners must align to the top. If the frame is viewed from the incorrect side, the colors are given a red tint to indicate that the frame is facing the wrong direction. The position and orientation of the navigational frames presented to the user differ depending on the input device used. This was done partially to decrease the learnability of this particular phase, but also because the interaction styles differ fundamentally. The mouse interface essentially rotates a camera around a given viewpoint (in this case, the pig carcass), where as the Phantom Omni allows the user easier access to look at regions of no interest.

A pilot test performed prior to the actual usability study revealed the navigation test to have become the subject of developer difficulty. In other words, the corners were much too difficult to align properly, meaning participants would quickly have the frame oriented correctly, but getting each corner to fit took almost as much time as orienting the frame initially. The sections within which the corners had to align were made much bigger as a result.

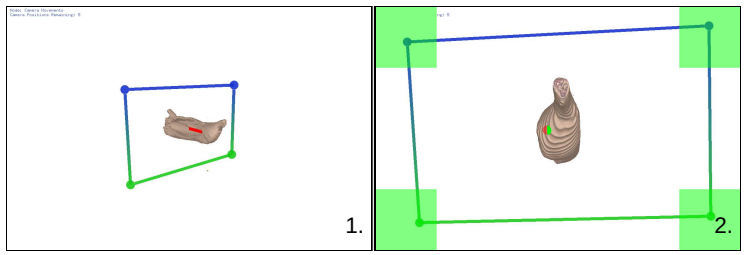

Figure 11.8: Screenshots of a single navigational frame during navigation practice. 1. An example of what the user initially sees during the navigational challenge. 2. Aligning the frame to fit the view window with the green corners at the bottom and blue corners at the top completes a single navigational task.

11.6.2.2 Product Creation

Each volunteer was asked to create five standardized products, pictured in Fig. 11.6, using the interface, as quickly and accurately as possible. Every participant had prior knowledge of the products and were provided photographs of the products in question. As none of the participants had any formal training, it is unrealistic to expect anything near a precise reproduction of the virtual products created using expert knowledge and visualized in Fig. 11.6. The focus is placed on potential usability issues, as well as how easy it is for the users to create the required cuts to recreate an approximate version of the product. The participant was asked tell when they were satisfied with the result, before the product was considered complete.

During preliminary testing, we contemplated the possibility of introducing live feedback in the product creation process, informing the user how accurate their own cutting planes were in relation to the expert version. However, this would not only have put too strong of an emphasis on the accuracy of the created product, but also indirectly pressured the user to define their own personal goals in regards to the expertly created product.

11.6.3 Results

Sauro and Lewis [105] have shown that the geometric mean is a better estimate of the middle task-time in usability studies, than the arithmetic mean or median, for a small sample size. With the limited number of available sales staff as volunteers, we use the geometric mean when reporting all of the central tendencies of task times. The error bars applied visualized in relation to task

times correspond to a 95% confidence interval as recommended by Streiner [111]. We also apply an ANOVA test to the collected data for a concrete numerical comparison between the two interfaces.

During the usability study, the PPP crashed on a single occasion while being used with the Phantom Omni by a test participant. Consequently, a small amount of data for this participant is unavailable. The confidence intervals are adjusted accordingly with respect to 7 or 8 samples given the results being calculated.

11.6.3.1 Applied Performance Metrics

The mean time for each navigational task completed using mouse and Phantom Omni is visualized in Fig. 11.9.

Visually, it's hard to discern any tendency in the data visualized in Fig. 11.9. To get a better sense of any significant tendency, we applied an analysis of variance [86] (ANOVA) which is described more in depth in the following section. The ANOVA confirmed our suspicion that there was no significant tendency for faster completion times with the Phantom Omni. As previously noted, the navigational challenges differ between interfaces and a direct comparison is therefore not ideal.

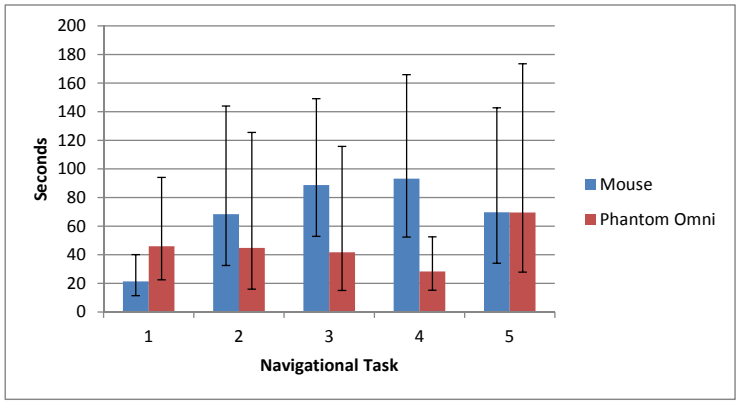

Figure 11.9: The geometric mean time taken by users to solve the individual navigation tasks using the mouse and the Phantom Omni. The error bars represent the 95% confidence interval using the geometric standard deviation based on a students t distribution.

The mean time taken to create each product is a more accurate basis for com-

parison of performance between the interfaces, and is visualized in Fig. 11.10. The ANOVA showed that there is no significant difference in task completion time between the mouse and the Phantom Omni.

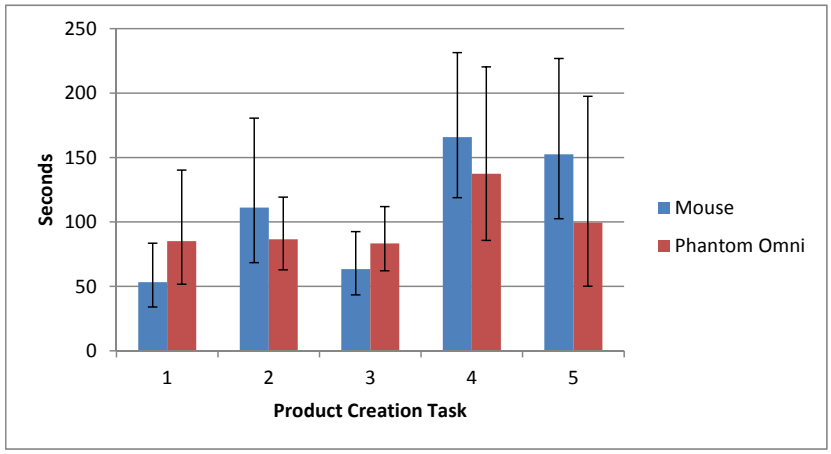

Figure 11.10: The geometric mean time taken by users to create each of the five meat products. The error bars represent the 95% confidence interval using the geometric standard deviation based on a students t distribution.

The quantitative comparison between user and expert meat products visualized in Fig. 11.11 is included for the sake of full disclosure. However, the accuracy of the products cannot be used as a measure to evaluate the success of either interface since none of the test participants possessed the technical expertise to reproduce the products to the specifications near the expert level.

Instead a subjective evaluation of each product was performed by us, comparing the products directly to the expertly created counterparts. As the results show in Fig. 11.11, every user created product differs, on average, at least 10% from the expert version. However, all of the products had cuts reminiscent of the expert version despite less accurate placement, and were therefore deemed successful.

11.6.3.2 Interface ANOVA

We perform a 3-way analysis of variance (ANOVA) [86] (mixed effect) with pairwise interactions, where the interfaces and product creation tests are fixed effects and users are considered a random effect. As previously mentioned, we intentionally do not compare the navigational tests as they differ by design, and serve as part of the learning introduction.

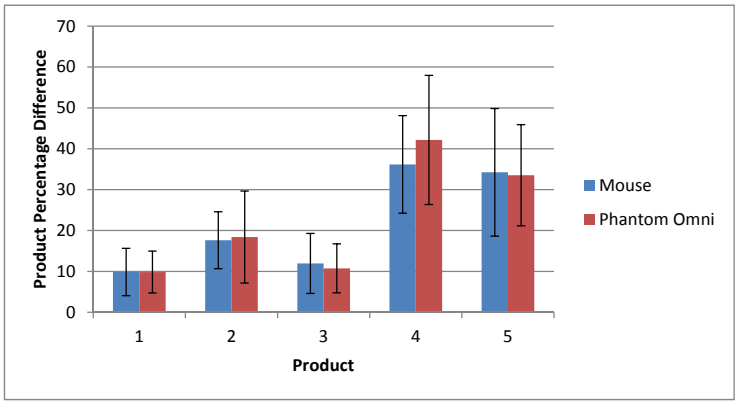

Figure 11.11: The percentage difference of the product created by the user relative to the product created using expert knowledge. An example of a product that is 100% different, could in this case be a product that is twice the size of the expert version.

The results, displayed in table 11.2, show no significant effect from the interface. Unsurprisingly, the analysis indicates that certain products are significantly easier/faster to create than others. It is also clear that there is a significant difference in performance with the interfaces per specific user.

Source	Sum Sq.	d.f.	Mean Sq.	F	Prob>F
User	5.3925	6	0.89876	1.49	0.3308
Interface	0.0001	1	0.00006	0	0.9928
Test no.	5.5283	4	1.38208	8.56	0.0002
User*Interface	3.7328	6	0.62214	3.48	0.0129
User*Test no.	3.8732	24	0.16138	0.9	0.599
Interface* Test no.	4.2952	4	0.56971	3.18	0.0312
Error	4.2952	24	0.17897	0	0
Total	25.1009	69			

Table 11.2: Results from the analysis of variance with pairwise interaction, where interfaces and products are fixed, and with users considered a random effect.

We also performed an ANOVA where product creation tests were considered a random effect rather than fixed, which produced similar results for the interfaces.

11.6.3.3 SUS Questionnaire

The SUS score was calculated according to method presented by Brookes [16]. The score on its own, while still an evaluation of a given system, is without meaning without a relation to other scores. Bangor et al. [9, 10] have analyzed a large body of work using the SUS, and established an overall grading scale for evaluating scores. According to this established scale, both interfaces are mediocre (or OK).

Interface	Mouse	Phantom Omni
Mean SUS Score	55,3125	56,875
Lower 95% Confidence	46,846	46,898
Upper 95% Confidence	63,779	66,852

Table 11.3: The mean SUS score for each interface along with upper and lower 95% confidence.

11.6.3.4 Observations

We noted the following useful observations during the course of the comparative usability study:

- **User fatigue** - A single case of user fatigue was noted during testing with the Phantom Omni interface. While an isolated case, we believe the significance of this should not be overlooked. For the Phantom Omni to be optimally incorporated into any daily use, it will require its users to acclimate themselves to become fairly agile with their wrists. The age of the user and by extension the expected health of the users various muscle groups have a more significant impact in regards to the Phantom Omni, in our opinion. Zhai et al. [133] also reported a case of user fatigue in their study of muscle groups affecting performance.

- **Plane Selection** - On a few occasions, test participants experienced difficulty in selecting a single isolated plane. We considered solutions for each of the interfaces to overcome this issue. In the case of the mouse interface, the thickness of the cutting plane frames could be adjusted to fit the number of nearby planes to allow for easier selection when they are isolated. In the case of the Phantom Omni interface, a direct line in between the cursor and the nearest plane might help mitigate issues with perceived depth.

- **Control Confusion** - Users would occasionally mistake the function of one of the buttons with that of another, both in the case of the mouse as well as the Phantom Omni interface. Additional visual and audial cues might help users better remember which buttons are assigned which functions as well as longer, more tutorialized, introductions.
- **Navigation Plane** - Three users initially tried to use the cursor to grab the navigational planes during the navigational testing phase. Making them less similar to that of the cutting planes would likely resolve this misunderstanding.
- **Phantom Omni Continuous Button Holding** - During preliminary testing we experienced occasional problems in holding down the Phantom Omnis buttons continuously. We also noticed similar issues during both pilot tests as well as during the usability study. We believe this to primarily be the result of low quality buttons on the Phantom Omni.
- **Haptic Feedback** - Haptic feedback has previously been shown to improve accuracy, but not task times, in 3D target acquisitional tasks [118]. None of the test participants seem to particularly notice the magnetic draw to the cut planes, nor was the haptic feedback provided by the pig used extensively.
- **Overall Preference** - Five of the eight participants expressed to prefer the Phantom Omni. Given the nearly identical SUS score, we believe that a significant part of the users preference is based on anticipated performance, given more time with the interface.
- **Phantom Omni Preference** - Almost all of the users who noted their preference for the Phantom Omni, noted it as being easier to navigate three dimensional space with. A single user noted the preference due to all of the necessary functions being operable from one hand.
- **Mouse Preference** - Out of the three users preferring the mouse, two noted that with additional experience they would have preferred the Phantom Omni. A single user noted preference for the mouse based on ease of learning and compact size for mobility.

11.7 Conclusion

The formative usability study of the PPP interface provided valuable information revealing minor usability issues affecting both interfaces. As for the usability of both interfaces, the applied metrics, SUS scores, and ANOVA do

not lean with any significance in favor of either the mouse or the Phantom Omni interface. The significance of these results come from the fact that both interfaces were introduced in a very short time frame, and that none of the test participants had had any previous experience with an input controller resembling the Phantom Omni. Neither of these factors impacted the performance with the Phantom Omni in such a fashion as to exhibit worse performance than the mouse interface.

The ease with which an interface is learned has an impact on how readily a user accepts its integration into their daily working routine, and despite the daily use of the mouse, the Phantom Omni was able to deliver similar results. It is of course important to also emphasize, that none of the test participants had had extensive practice with an interface similar to the mouse cutting interface in the PPP. The Phantom Omni also provides four additional degrees of freedom, making it arguably easier to use in a three dimensional virtual space.

It is our belief that in the case of this and similar applications, the Phantom Omni is a worthwhile avenue of exploration as substitution for the mouse. However, a longer introductory period is required to in order to become sufficiently proficient with it.

11.7.1 Future Work

The SUS scores reveal that there is still significant room for improvement in both interfaces. The following points are what we believe to be the most interesting avenues of improvement regarding the PPP application:

- **Interactive Tutorials** - Tutorials visualizing the connection between specific input controller use and effect on the virtual scene would likely accelerate the learning process. Due to the almost infinite number of tutorial design possibilities, it would be interesting to measure how effective which tutorials would be at improving the SUS score without altering the interface.

- **Expanded Testing Group** - Expert knowledge is not a pre-requisite for proper use of the PPP application. Therefore, usability issues that are harder to spot could be uncovered by increasing the number of participants in subsequent tests. Tullis and Albert [115] also note the necessity to increase the number of participants in a test as a system or product is close to being refined and finished.

- **Prototyping Improvements** - All the feedback gathered during the usability study and subsequent interviews is highly valuable and should drive future design. During an expert interview, we were told that membranes separating muscle groups are frequently used to perform more specialized cuts. Not only do these type of separational cuts ease the handling of the meat, but they are also unambiguous compared to the early planar cuts imposed on the pig carcasses. Implementing these types of cuts would make the PPP more useful during as a product prototyper. It would also be beneficial to generalize the cuts to any conceivable variation in the pigs anatomy using a so-called pig atlas [47].
- **Expert Knowledge Embedding** - The PPP has the potential to not only supplant static imagery, but also imbue even the most novice user with expert knowledge, by providing constructive feedback regarding created cuts. Such feedback ranges from anything in between physical and fiscal feasibility of cuts, to resulting byproducts.

11.8 Acknowledgments

We would like to thank the DMRI [28] and Danish Crown for their collaboration during the development of PPP. We would also like to thank the Tokyo University for acting as partial host during the development period. The collaborative effort between universities was partially funded by Otto Mønsted Fonden, Augustinus Fonden, and the Japan-Scandinavian Sasakawa Foundation who we thank. We also thank the volunteers who partook in the usability test, making this research possible. Finally, we'd like to thank Kasper Hornbæk for his feedback during the development of this paper.

Part III

Appendix

List of Figures

Bibliography

[1] Cgtextures. http://www.cgtextures.com/, 2011.

[2] 2textured. http://www.2textured.com/, 2012.

[3] Archive textures. http://archivetextures.net/, 2012.

[4] Grunge textures. http://www.grungetextures.com/, 2012.

[5] Mayang's free textures version 15. http://mayang.com/textures/, 2012.

[6] Texture king. http://www.textureking.com/, 2012.

[7] M. J. Ackerman. The visible human project: A resource for education. *Acad. Med.*, 74:667 – 670, 1999.

[8] Arthur Appel. Some techniques for shading machine renderings of solids. In *Proceedings of the April 30–May 2, 1968, spring joint computer conference*, AFIPS '68 (Spring), pages 37–45, New York, NY, USA, 1968. ACM.

[9] A. Bangor, P. Kortum, and J. Miller. Determining what individual sus scores mean: Adding an adjective rating scale. *Journal of usability studies*, 4(3):114–123, 2009.

[10] A. Bangor, P.T. Kortum, and J.T. Miller. An empirical evaluation of the system usability scale. *Intl. Journal of Human–Computer Interaction*, 24(6):574–594, 2008.

[11] C. Barnes, E. Shechtman, A. Finkelstein, and D.B. Goldman. Patchmatch: a randomized correspondence algorithm for structural image editing. In *ACM Transactions on Graphics (TOG)*, volume 28, page 24. ACM, 2009.

[12] E. C. Beckmann. CT scanning the early days. *British Journal of Radiology*, 79(937):5–8, January 2006.

[13] G. Bernstein and D. Fussell. Fast, exact, linear booleans. In *Computer Graphics Forum*, volume 28, pages 1269–1278. Wiley Online Library, 2009.

[14] B.T. Bethea, A.M. Okamura, M. Kitagawa, T.P. Fitton, S.M. Cattaneo, V.L. Gott, W.A. Baumgartner, and D.D. Yuh. Application of haptic feedback to robotic surgery. *Journal of Laparoendoscopic & Advanced Surgical Techniques*, 14(3):191–195, 2004.

[15] P. Brodatz. *Textures: a photographic album for artists and designers*, volume 66. Dover New York, 1966.

[16] J. Brooke. Sus: A "quick and dirty" usability scale. *P. W. Jordan, B. Thomas, B. A. Weerdmeester, & A. L. McClelland (Eds.) Usability Evaluation in Industry*, pages 189–194, 1996.

[17] C.D. Bruyns, S. Senger, A. Menon, K. Montgomery, S. Wildermuth, and R. Boyle. A survey of interactive mesh-cutting techniques and a new method for implementing generalized interactive mesh cutting using virtual tools. *The journal of visualization and computer animation*, 13(1):21–42, 2002.

[18] P. Böttcher, J. Maierl, T. Schiemann, C. Glaser, R. Weller, K.H. Hoehne, M. Reiser, and H.G. Liebich. The visible animal project: A three-dimensional, digital database for high quality three-dimensional reconstructions. *Veterinary Radiology & Ultrasound*, 40(6):611–616, 1999.

[19] Jesus J. Caban and Penny Rheingans. Texture-based transfer functions for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1364–1371, 2008.

[20] D. Chang. Haptics: gaming's new sensation. *Computer*, 35(8):84–86, 2002.

[21] K.W. Chen, P.A. Heng, and H. Sun. Direct haptic rendering of isosurface by intermediate representation. In *Proceedings of the ACM symposium on Virtual reality software and technology*, pages 188–194. ACM, 2000.

[22] Michael Chen, S. Joy Mountford, and Abigail Sellen. A study in interactive 3-d rotation using 2-d control devices. *SIGGRAPH Comput. Graph.*, 22(4):121–129, June 1988.

[23] A. Chourasia and J.P. Schulze. Data centric transfer functions for high dynamic range volume data. In *Proc. International Conf. Computer Graphics, Visualization, and Computer Vision*, 2007.

[24] Line Katrine Harder Clemmensen and Lasse Farnung Laursen. Improving texture optimization with application to visualizing meat products. IMM-Technical Report-Â2011 ; 15, pages 81–86, Kgs. Lyngby, Denmark, 2011. Technical University of Denmark. Presented at: Scandinavian Workshop on Imaging Food Quality, SWIFQ : Ystad, Sweden, 2011.

[25] Timothy J. Cullip and Ulrich Neumann. Accelerating volume reconstruction with 3d texture hardware. Technical report, Chapel Hill, NC, USA, 1994.

[26] DC. Danish crown. http://www.danishcrown.dk/, April 2012.

[27] Jeremy S. De Bonet. Multiresolution sampling procedure for analysis and synthesis of texture images. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 361–368, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[28] DMRI. Danish meat research institute. http://www.teknologisk.dk/dmri, April 2012.

[29] Feng Dong and Gordon J. Clapworthy. Volumetric texture synthesis for non-photorealistic volume rendering of medical data. *The Visual Computer*, 21:463–473, 2005.

[30] Yue Dong, Sylvain Lefebvre, Xin Tong, and George Drettakis. Lazy solid texture synthesis. In *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering)*, 2008.

[31] Robert A. Drebin, Loren Carpenter, and Pat Hanrahan. Volume rendering. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pages 65–74, New York, NY, USA, 1988. ACM.

[32] Søren Gylling Hemmingsen Erbou. *Modeling the Biological Diversity of Pig Carcasses*. PhD thesis, Technical University of Denmark, Department of Informatics and Mathematical Modeling, Image Analysis and Computer Graphics, Kgs. Lyngby, Denmark, feb 2009.

[33] Landbrug & Fødevarer. Landburgsvareeksporten. http://www.lf.dk/Tal_og_Analyser/Aktuelle_statistikker/~/media/lf/Tal%20og%20analyser/Aktuelle%20statistikker/eksportstatistik/Landbrugsvareeksporten1111.ashx, February 2012.

[34] Landbrug & Fødevarer. Slagtninger af svin i danmark. http://www.lf.dk/Tal_og_Analyser/Aktuelle_statistikker/Svin/slagtninger.aspx, February 2012.

[35] Landbrug & Fødevarer. Statistics 2010 pigmeat. http://www.lf.dk/Tal_og_Analyser/Aarstatistikker/Statistik_svin/~/media/lf/Tal%20og%20analyser/Aarsstatistikker/Statistik%20svin/2010/108-2011_A5%20Statistik%20UK2010_WEB.ashx, February 2012.

[36] David H. Frakes, Lakshmi P. Dasi, Kerem Pekkan, Hiroumi D. Kitajima, Kartik Sundareswaran, Ajit P. Yoganathan, and Mark J. T. Smith. A new method for registration-based medical image interpolation. *IEEE Trans. Med. Imaging*, 27(3):370–377, 2008.

[37] T.A. Galyean and J.F. Hughes. Sculpting: An interactive volumetric modeling technique. In *ACM SIGGRAPH Computer Graphics*, volume 25, pages 267–274. ACM, 1991.

[38] Vincent Garcia, Eric Debreuve, Frank Nielsen, and Michel Barlaud. k-nearest neighbor search: fast GPU-based implementations and application to high-dimensional feature matching. In *IEEE International Conference on Image Processing (ICIP)*, Hong Kong, China, September 2010.

[39] D. Ghazanfarpour and J. M. Dischler. Spectral analysis for automatic 3-d texture generation. *Computers & Graphics*, 19(3):413 – 422, 1995.

[40] Bjørn Godske. Slagterirobot skal skære lige til benet. *Ingeniøren*, 2008. http://ing.dk/artikel/88774-slagterirobot-skal-skaere-lige-til-benet.

[41] L.W. Goldman. Principles of ct and ct technology. *Journal of nuclear medicine technology*, 35(3):115–128, 2007.

[42] A.D. Gregory, S.A. Ehmann, and M.C. Lin. intouch: Interactive multiresolution modeling and 3d painting with a haptic interface. In *Virtual Reality, 2000. Proceedings. IEEE*, pages 45–52. IEEE, 2000.

[43] G. J. Grevera and J. K. Udupa. An Objective Comparison of 3-D Image Interpolation Methods. *IEEE Transactions on Medical Imaging*, 17(4):642–652, August 1998.

[44] Markus Hadwiger, Joe M. Kniss, Christof Rezk-salama, Daniel Weiskopf, and Klaus Engel. *Real-time Volume Graphics*, pages 81–102. A. K. Peters, Ltd., Natick, MA, USA, 2006.

[45] Markus Hadwiger, Joe M. Kniss, Christof Rezk-salama, Daniel Weiskopf, and Klaus Engel. *Real-time Volume Graphics*, pages 163–185. A. K. Peters, Ltd., Natick, MA, USA, 2006.

[46] Markus Hadwiger, Joe M. Kniss, Christof Rezk-salama, Daniel Weiskopf, and Klaus Engel. *Real-time Volume Graphics*, pages 114–116. A. K. Peters, Ltd., Natick, MA, USA, 2006.

[47] Mads Fogtmann Hansen. Designing and analyzing virtual cuts in 3d models of pig bodies by mapping cuts from a statistical atlas. Presented at: European congress of Chemical Engineering : Bella Center, Copenhagen, 2006.

[48] Mads Fogtmann Hansen. *The virtual knife*. PhD thesis, Technical University of Denmark, Department of Informatics and Mathematical Modeling, Image Analysis and Computer Graphics, Kgs. Lyngby, jul 2009.

[49] M. Harders, A. Barlit, K. Akahane, M. Sato, and G. Székely. Comparing 6dof haptic interfaces for application in 3d assembly tasks. In *Proc. of EuroHaptics*, July 2006.

[50] B.D. Harper and K.L. Norman. Improving user satisfaction: The questionnaire for user interaction satisfaction version 5.5. In *Proceedings of the 1st Annual Mid-Atlantic Human Factors Conference*, pages 224–228, 1993.

[51] P. Harrison. A non-hierarchical procedure for re-synthesis of complex textures. In *W S C G ' 2001, VOLS I & II, CONFERENCE PROCEEDINGS*, 2001.

[52] David J. Heeger and James R. Bergen. Pyramid-based texture analysis/synthesis. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 229–238, New York, NY, USA, 1995. ACM.

[53] Lars Hinrichsen. Manufacturing technology in the danish pig slaughter industry. *Meat Science*, 84(2):271 – 275, 2010. ¡ce:title¿Special Issue: 55th International Congress of Meat Science and Technology (55th ICoMST), 16-21 August 2009, Copenhagen, Denmark¡/ce:title¿.

[54] B.W. Hong, S. Soatto, K. Ni, and T. Chan. The scale of a texture and its application to segmentation. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[55] B. Itkowitz, J. Handley, and W. Zhu. The openhaptics toolkit: a library for adding 3d touch navigation and haptics to graphics applications. In *Eurohaptics Conference, 2005 and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2005. World Haptics 2005. First Joint*, pages 590–591. IEEE, 2005.

[56] Robert Jagnow, Julie Dorsey, and Holly Rushmeier. Stereological techniques for solid textures. *ACM Trans. Graph.*, 23(3):329–335, 2004.

[57] Nielsen J.U, Fertin C., and Christensen H. Up-to-date equipment for pig slaughtering, cutting and boning and their influence on product safety. *Tehnologija mesa*, 46(1-2):62–66, 2005.

[58] Willi Kalender and Kevin O Khadivi. Computed tomography: Fundamentals, system technology, image quality, applications, 2nd edition. *Medical Physics*, 33(8):3076–3076, 2006.

[59] David Victor Keyson. *Touch In User Interface Navigation*. PhD thesis, Technical University of Eindhoven, 1996.

[60] J. Kirakowski and M. Corbett. Sumi: The software usability measurement inventory. *British journal of educational technology*, 24(3):210–212, 1993.

[61] Niels Christian Kjærsgaard. *Optimization of the raw material use at Danish slaughterhouses*. PhD thesis, Technical University of Denmark, Department of Management Engineering, Operations Research, sep 2008.

[62] Johannes Kopf, Chi-Wing Fu, Daniel Cohen-Or, Oliver Deussen, Dani Lischinski, and Tien-Tsin Wong. Solid texture synthesis from 2d exemplars. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2007)*, 26(3):2:1–2:9, 2007.

[63] Johannes Kopf, Chi-Wing Fu, Daniel Cohen-Or, Oliver Deussen, Dani Lischinski, and Tien-Tsin Wong. Solid texture synthesis from 2d exemplars. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2007)*, 26(3):2:1–2:9, 2007.

[64] Ross Koppel, Joshua P. Metlay, Abigail Cohen, Brian Abaluck, A. Russell Localio, Stephen E. Kimmel, and Brian L. Strom. Role of computerized physician order entry systems in facilitating medication errors. *JAMA: The Journal of the American Medical Association*, 293(10):1197–1203, 2005.

[65] Jens Krüger and Rüdiger Westermann. Acceleration techniques for gpu-based volume rendering. In *IEEE Visualization*, pages 287–292, 2003.

[66] Vivek Kwatra, Irfan Essa, Aaron Bobick, and Nipun Kwatra. Texture optimization for example-based synthesis. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, pages 795–802, New York, NY, USA, 2005. ACM.

[67] Vivek Kwatra, Arno Schödl, Irfan Essa, Greg Turk, and Aaron Bobick. Graphcut textures: image and video synthesis using graph cuts. *ACM Trans. Graph.*, 22:277–286, July 2003.

[68] P. Lacroute and M. Levoy. Fast volume rendering using a shear-warp factorization of the viewing transformation. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 451–458. ACM, 1994.

[69] H. Ólafsdottir, H. Pedersen, M. S. Hansen, M. Lyksborg, S. Darkner, and R. Larsen. Registration-based interpolation applied to cardiac mri. In *proceedinsg of SPIE - International Society for Optical Engineering, International Symposium on Medical Imaging 2010*, 2010.

[70] Jakob Andreas Bærentzen Takeo Igarashi Bjarne Kjær Ersbøll Lasse Farnung Laursen, Line Harder Clemmensen. Pig product prototyper: Cutting interface design. *NordiCHI 2012 Proceedings*.

[71] Jakob Andreas Bærentzen Takeo Igarashi Bjarne Kjær Ersbøll Lasse Farnung Laursen, Line Harder Clemmensen. Automatic quality measurement and parameter selection for example-based texture synthesis. Technical Report IMM-Technical-Report-2012-07, Technical University of Denmark, Department of Informatics and Mathematical Modeling, April 2012.

[72] Lasse Laursen, Bjarne Kjær Ersbøll, and Jakob Andreas Bærentzen. Anisotropic 3d texture synthesis with application to volume rendering. In *Proceedings of WSCG'2011 (19-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision'2011)*, 2011.

[73] Lasse Farnung Laursen and Bjarne Kjær Ersbøll. Gazetrain: A case study of an action oriented gaze-controlled game. pages 61–65, Kgs. Lyngby, Denmark, 2009. Technical University of Denmark. Presented at: The 5th Conference on Communication by Gaze Interaction - COGAIN 2009: Gaze Interaction For Those Who Want It Most.

[74] Lasse Farnung Laursen, Bjarne Kjær Ersbøll, and Jakob Andreas Bærentzen. Anisotropic 3d texture synthesis with application to volume rendering. In *WSCG' 2011 Communication Papers Proceedings*, pages 49–57, 2011. Presented at: International Conference on Computer Graphics, Visualization and Computer Vision, WSCG ; 19 : Plzen, Czech Republic, 2011.

[75] Lasse Farnung Laursen, Hildur Ólafsdottir, Jakob Andreas Bærentzen, Michael Sass Hansen, and Bjarne Kjær Ersbøll. Registration-based interpolation real-time volume visualization. In *Proceedings of the 28th Spring Conference on Computer Graphics*. ACM, 2012.

[76] SD Laycock and AM Day. A survey of haptic rendering techniques. In *Computer Graphics Forum*, volume 26, pages 50–65. Wiley Online Library, 2007.

[77] Sylvain Lefebvre and Hugues Hoppe. Parallel controllable texture synthesis. *ACM TRANSACTIONS ON GRAPHICS*, pages 777–786, 2005.

[78] W.E. Lorensen and H.E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *ACM Siggraph Computer Graphics*, volume 21, pages 163–169. ACM, 1987.

[79] Aidong Lu, David S. Ebert, Wei Qiao, Martin Kraus, and Benjamin Mora. Volume illustration using wang cubes. *ACM Trans. Graph.*, 26, June 2007.

[80] Sean Luke. *Essentials of Metaheuristics*. Lulu, 2009. Available for free at http://cs.gmu.edu/∼sean/book/metaheuristics/.

[81] A.M. Lund. Measuring usability with the use questionnaire. *Usability and User Experience*, 8(2):8, 2001.

[82] Felix Manke and Burkhard C. Wuensche. Texture-enhanced direct volume rendering. In *Proceedings of the 4th International Conference on Computer Graphics Theory and Applications (GRAPP 2009)*, pages 185–190, Lisbon, Portugal, 2009.

[83] Felix Manke and Burkhard Wunsche. Fast three-dimensional texture synthesis. *New Zealand Computer Science Research Student Conference*, 2008.

[84] Erik Meijering. A chronology of interpolation: From ancient astronomy to modern signal and image processing. In *Proceedings of the IEEE*, pages 319–342, 2002.

[85] B. Menelas, M. Ammi, and P. Bourdot. A flexible method for haptic rendering of isosurface from volumetric data. *Haptics: Perception, Devices and Scenarios*, pages 687–693, 2008.

[86] R.G. Miller and B.W. Brown. *Beyond ANOVA: basics of applied statistics*. Chapman & Hall/CRC, 1997.

[87] Tobias Moench, Simon Adler, and Bernhard Preim. Staircase-aware smoothing of medical surface meshes. In *Eurographics Workshop on Visual Computing for Biology and Medicine (VCBM)*, 2010.

[88] Thomas Hammershaimb Mosbech. *Computed Tomography in the Modern Slaughterhouse*. PhD thesis, Technical University of Denmark, Department of Informatics and Mathematical Modeling, Image Analysis and Computer Graphics, Kgs. Lyngby, Denmark, 2011.

[89] David M. Mount and Sunil Arya. Ann: A library for approximate nearest neighbor searching. http://www.cs.umd.edu/ mount/ANN/, 2010.

[90] Ken Naono, Keita Teranishi, John Cavazos, and Reiji Suda, editors. *Software Automatic Tuning: From Concepts to State-of-the-Art Results*. Springer, 1st edition. edition, 9 2010.

[91] National Electrical Manufacturers Association. *Digital Imaging and Communications in Medicine (DICOM)*, 3.1-2011 edition, 2011.

[92] J. Nickolls and W.J. Dally. The gpu computing era. *Micro, IEEE*, 30(2):56–69, 2010.

[93] Jakob Nielsen. Medical usability: How to kill patients through bad design. http://www.useit.com/alertbox/20050411.html, February 2012.

[94] P. Omni. Technical specifications, 2007.

[95] K.L. Palmerius. *Direct Volume Haptics for Visualization*. PhD thesis, Department of Science and Technology, Linköpings universitet, 2007.

[96] G. P. Penney, J. A. Schnabel, D. Rueckert, M. A. Viergever, and W. J. Niessen. Registration-based interpolation. *IEEE transactions on medical imaging*, 23(7):922–926, July 2004.

[97] Ken Perlin. Noise hardware. In Real-Time Shading SIGGRAPH Course Notes (2001), Olano M., (Ed.)., 2001.

[98] A. Petersik, B. Pflesser, U. Tiede, K.H. Höhne, and R. Leuwer. Realistic haptic interaction in volume sculpting for surgery simulation. *Surgery Simulation and Soft Tissue Modeling*, pages 1001–1001, 2003.

[99] Beryl Plimmer. Experiences with digital pen, keyboard and mouse usability. *Journal on Multimodal User Interfaces*, 2(1):13–23, July 2008.

[100] J.D. Power and Associates. J.d. power and associates reports: The right blend of design and technology is critical to creating an exceptional user experience with smartphones and traditional mobile devices. Technical report, J.D. Power and Associates, 2011.

[101] Jeffrey Rubin and Dana Chisnell. *Handbook of Usability Testing: Howto Plan, Design, and Conduct Effective Tests, 2nd Edition*. Wiley, 2 edition, May 2008.

[102] Y. Rubner, C. Tomasi, and L.J. Guibas. A metric for distributions with applications to image databases. In *Computer Vision, 1998. Sixth International Conference on*, pages 59–66. IEEE, 1998.

[103] D. Rueckert, L.I. Sonoda, C. Hayes, D.L.G. Hill, M.O. Leach, and D.J. Hawkes. Nonrigid registration using free-form deformations: application to breast mr images. *Medical Imaging, IEEE Transactions on*, 18(8):712 –721, aug. 1999.

[104] J.H. Saltzer and F. Kaashoek. *Principles of computer system design: an introduction*. Morgan Kaufmann, 2009.

[105] J. Sauro and J.R. Lewis. Average task times in usability tests: what to report? In *Proceedings of the 28th international conference on Human factors in computing systems*, pages 2347–2350. ACM, 2010.

[106] Sensable. The freeform® systems. http://www.sensable.com/products-freeform-systems.htm, April 2012.

[107] D. Shreiner, M. Woo, J. Neider, and T. Davis. *OpenGL(R) Programming Guide : The Official Guide to Learning OpenGL(R), Version 2 (5th Edition)*. Addison-Wesley Professional, August 2005.

[108] D. Simakov, Y. Caspi, E. Shechtman, and M. Irani. Summarizing visual data using bidirectional similarity. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[109] M. Smithson. Confidence intervals: Quantitative applications in the social sciences series, no. 140, 2003.

[110] N.T. Stewart. *An Image-Space Algorithm for Hardware-Based Rendering of Constructive Solid Geometry*. PhD thesis, RMIT University, 2008.

[111] D.L. Streiner et al. Maintaining standards: differences between the standard deviation and standard error, and when to use each. *Canadian journal of psychiatry. Revue canadienne de psychiatrie*, 41(8):498, 1996.

[112] G.T. Sung and I.S. Gill. Robotic laparoscopic surgery: a comparison of the da vinci and zeus systems. *Urology*, 58(6):893–898, 2001.

[113] El-Ghazali Talbi. *Metaheuristics: From Design to Implementation (Wiley Series on Parallel and Distributed Computing)*. Wiley, 6 2009.

[114] U. Tiede, T. Schiemann, and K.H. Hohne. High quality rendering of attributed volume data. In *Visualization'98. Proceedings*, pages 255–262. Ieee, 1998.

[115] Thomas Tullis and William Albert. *Measuring the User Experience: Collecting, Analyzing, and Presenting Usability Metrics (Interactive Technologies)*. Morgan Kaufmann, March 2008.

[116] Martin Vester-Christensen. *Image Registration and Optimization in the Virtual Slaughterhouse*. PhD thesis, Technical University of Denmark, Department of Informatics and Mathematical Modeling, Â, Kgs. Lyngby, Denmark, feb 2009.

[117] Martin Vester-Christensen, SÃ¸ren Gylling Hemmingsen Erbou, Mads Fogtmann Hansen, E.V. Olsen, L.B. Christensen, M. Hviid, Bjarne Kjær Ersbøll, and Rasmus Larsen. Virtual dissection of pig carcasses. *Meat Science*, 81(4):699–704, 2009.

[118] S.A. Wall, K. Paynter, A.M. Shillito, M. Wright, and S. Scali. The effect of haptic feedback and stereo graphics in a 3d target acquisition task. In *Proceedings of eurohaptics*, pages 23–28, 2002.

[119] H. Wang. Proving theorems by pattern recognition i. *Communications of the ACM*, 3(4):220–234, 1960.

[120] H. Wang. Games, logic and computers. *Scientific American*, 213(5):98–106, 1965.

[121] S.W. Wang and A.E. Kaufman. Volume sculpting. In *Proceedings of the 1995 symposium on Interactive 3D graphics*, pages 151–ff. ACM, 1995.

[122] Yajun Wang, Jiaping Wang, Nicolas Holzschuch, Kartic Subr, Jun-Hai Yong, and Baining Guo. Real-time rendering of heterogeneous translucent objects with arbitrary shapes. *Computer Graphics Forum*, 29:497–506, April 2010.

[123] Li-Yi Wei. *Texture synthesis by fixed neighborhood searching*. PhD thesis, Stanford, CA, USA, 2002. Adviser-Levoy, Marc.

[124] Li-Yi Wei, Jianwei Han, Kun Zhou, Hujun Bao, Baining Guo, and Heung-Yeung Shum. Inverse texture synthesis. *ACM Trans. Graph.*, 27:52:1–52:9, August 2008.

[125] Li-Yi Wei, Sylvain Lefebvre, Vivek Kwatra, and Greg Turk. State of the art in example-based texture synthesis. In *Eurographics 2009, State of the Art Report, EG-STAR*. Eurographics Association, 2009.

[126] Li-Yi Wei and Marc Levoy. Order-independent texture synthesis. Earlier version is Stanford University Computer Science TR-2002-01.

[127] Li-Yi Wei and Marc Levoy. Fast texture synthesis using tree-structured vector quantization. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 479–488, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[128] Yonatan Wexler, Eli Shechtman, and Michal Irani. Space-time completion of video. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(3):463–476, 2007.

[129] Wikipedia. Bicubic interpolation — Wikipedia, the free encyclopedia, 2012. [Online; accessed 11-April-2012].

[130] Wikipedia. Line-plane intersection — Wikipedia, the free encyclopedia, 2012. [Online; accessed 09-April-2012].

[131] Wikipedia. Simulated annealing — Wikipedia, the free encyclopedia, 2012. [Online; accessed 18-April-2012].

[132] Qing Wu and Yizhou Yu. Feature matching and deformation for texture synthesis. *ACM Trans. Graph.*, 23:364–367, August 2004.

[133] Shumin Zhai, Paul Milgram, and William Buxton. The influence of muscle groups on performance of multiple degree-of-freedom input. In *Proceedings of the SIGCHI conference on Human factors in computing systems: common ground*, CHI '96, pages 308–315, New York, NY, USA, 1996. ACM.

[134] Weihang Zhu and Yuan-Shin Lee. Five-axis pencil-cut planning and virtual prototyping with 5-dof haptic interface. *Computer-Aided Design*, 36(13):1295–1307, 2004.